DESIGN AND EVALUATION OF A RELIABLE P2P COMMUNICATION PROTOCOL FOR IEEE 802.11B NETWORKS

BY

IRFAN ALI KHAN

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER ENGINEERING

JANUARY 2011

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS

DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **IRFAN ALI KHAN** under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER ENGINEERING**.

Thesis Committee

Dr. Mayez Al-Mouhamed (Advisor)

Dr. Ashraf S. H. Mahmoud (Member)

Dr. Zubair Ahmed Baig (Member)

Dr. Basem Al- Madani
Department Chairman

Dr. Salam A. Zummo
Dean of Graduate Studies

12/3/11

Date

Dedicated to

My Mother

And

My Family Members

ACKNOWLEDGEMENTS

All praises, all glory and all thanks are due to Allah, The Majestic, The Almighty for bestowing me with knowledge, guidance, patience, courage and health to achieve this work. May peace and blessings be upon prophet Mohammed (PBUH), his family and his companions.

I would like to acknowledge KFUPM for the support extended towards my research through its remarkable facilities and for providing me the opportunity to pursue graduate studies.

My deepest appreciation to my thesis advisor Dr. Mayez Al-Mouhamed for his constant endeavor, guidance, positive criticism and the numerous moments of attention, he devoted throughout this research work. His valuable suggestions made this work interesting and knowledgeable for me. I extend my thanks for all the committee members for their constructive support and encouragement.

I also owe thanks and recognition to my fellow RAs, course mates, colleagues and hostel friends for their help, motivation and support. Not all the names are possible to be mentioned here, but each one of them really made my stay at KFUPM joyful and memorable for lifetime. Special thanks to my seniors – Abdurrahman Bhai, Mohammed Siraj, Hasan Baig, Abu Faisal, Mumtaz Ahmed, Sarfaraz Ahmed, Khadir Khan, Faizan Ahmed and Murtuza Baig for their guidance and moral support throughout my stay at KFUPM. I am also obliged to thank all my friends especially Naeem and Rizwan for their

positive criticism, and Akber for all his support. Thanks to Minhaj, Abdul Malik, Nazeer Bhai, and Khaleel for their companionship.

Finally, from the bottom of my heart, I thank my mother and father, and all my family members for their continuous love, encouragement, prayers, emotional and moral support throughout my life. Words fall short in conveying my gratitude towards them. A prayer is the simplest way I can repay them – May Allah (S.W.T) give them good health and give me ample opportunity to be of service to them throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

APPENDIX

THESIS ABSTRACT (ENGLISH)

NAME: IRFAN ALI KHAN

TITLE: Design and Evaluation of a Reliable P2P Communication Protocol for IEEE 802.11b Networks

MAJOR: COMPUTER ENGINEERING

DATE: January 2011

The IEEE 802.11 based wireless local area networks (WLANs) have enjoyed popularity in many segments. In highly dynamic environments like Rescue missions or RoboCup, the communications among multiple robots are performed by broadcasting the data to their teammates called an Auction. Reliable, fast, and power aware communication is needed for these types of Ad-Hoc wireless networks. Broadcasting in IEEE 802.11 does not support any MAC layer recovery as a result due to interference and collisions in the wireless networks, this mechanism leaves broadcasting unreliable. Currently available techniques are based on Client-Server and Publish/Subscribe communication models, and are not suitable for multi-robot systems in which each robot acts like a peer. For this, we proposed reliable peer-to-peer communication protocols such as UDP P2P Reliable Broadcast with Token Passing and UDP P2P Reliable Distributed Broadcast. The protocols are implemented on a WLAN using the Stargate embedded system. The protocols are implemented using (1) a communication thread (TC) and (2) a processing thread (TP). A testbed system, which allows modules to run TC and TP, in addition to the generation of broadcast request, is presented in this thesis. Symmetric code is run in all the nodes (peer to peer communication). The evaluation of the protocols revealed : Response times are comparable to the UBTP auction scheme operated at head node, improved degree of reliability; at most 3 steps for seven nodes for UDP P2P Reliable Broadcast with Token Passing scheme, and at most 4 steps for seven nodes for UDP P2P Reliable Distributed Broadcast scheme. Comparable power consumption to simple UBTP auction scheme, slight variations in the average auction times when devices are mobile, symmetric performance pattern of all the nodes in both the proposed protocols.

MASTER OF SCIENCE DEGREE

KING FAHD UNIVERSITY OF PETROLEUM and MINERALS

Dhahran, Saudi Arabia

THESIS ABSTRACT (ARABIC)

الاسم : عرفان علي خان

العنوان : تصميم وتقييم لـP2P بروتوكول اتصال موثوقة لشبكات IEEE 802.11b

التخصص : هندسة الحاسب الآلي

التاريخ : يناير 2011

تتمتع الشبكات اللاسلكية المحلية (WLAN) المعتمدة على ميفاق IEEE 802.11 بشعبية كبيرة في العديد من القطاعات. في البيئات ذات الحركية العالية كمهام الإنقاذ أو مسابقة RoboCup فإن الاتصالات بين عدة روبوتات تجري من خلال بث البيانات إلى زملائهم في الفريق، وفي هذا النوع من الشبكات اللاسلكية غير المنتظمة فإننا بحاجة إلى اتصال موثوق وسريع ومدرك للطاقة المستهلكة. لا يدعم البث في ميفاق IEEE 802.11 أي عملية استعادة لطبقة MAC نتيجة التداخل والتصادمات في الشبكات اللاسلكية، وذلك ما يبقي عملية البث غير موثوقة. تعتمد الوسائل المتوفرة حالياً على نموذجي المخدم/العميل و النشر/الاشتراك، وهي لا تناسب النظم متعددة الروبوتات، والتي يتصرف فيها كل روبوت كنِدّ. لذلك اقترحنا موافيق اتصالات ند لند موثوقة مثل البث الموثوق لـ UDP P2P مع تمرير العلامة والبث الموزع الموثوق لـ UDP P2P . تنفّذ هذه الموافيق على شبكة محلية لاسلكية باستخدام نظام Stargate المضمن. تنفّذ هذه الموافيق باستخدام نيسب اتصال ونيسب معالجة. تقدم هذه الأطروحة نظام اختبار يسمح للوحدات بتنفيذ نيسبي الاتصال والمعالجة إضافةً إلى توليد طلب البث. ينفَّذ رماز متماثل على العقد كلها (اتصال ند لند). يكشف تقويم الموافيق عما يلي: أزمنة الاستجابة متوافقة مع نموذج المزاد لـUBTP العامل على العقدة الرئيسة، ودرجة أفضل من الوثوقية (3 خطوات على الأكثر من أجل 7 عقد في البث الموثوق لـUDP P2P مع تمرير العلامة، و 4 خطوات على الأكثر من أجل 7 عقد في البث الموزع الموثوق لـ UDP P2P)، وكذلك استهلاك الطاقة متقارب مع نموذج المزاد البسيط لـUBTP، واختلافات بسيطة في متوسط أزمنة المزاد عندما تكون الأجهزة متنقلة، ونمط أداء متماثل في العقد جميعها في كلا الميفاقين المقترحين.

درجة ماجستير علوم

جامعة الملك فهد للبترول والمعادن

الظهران، المملكة العربية السعودية

CHAPTER 1

INTRODUCTION

A wireless LAN (WLAN) is similar to a wired LAN but instead of using traditional wired structures, it uses radio waves as its transport medium. This allows the users to move around in a limited area while remaining connected to the network. Thus, WLANs enable movable LANs by combining data connectivity with user mobility through a simplified configuration. In other words, WLANs provide all the functionality of wired LANs, but without the physical constraints of the wire itself. The single most important feature of wireless networking that makes it dramatically different from wired networking is the wireless channel characteristic. The wireless transmission medium is inherently broadcast in nature. Many mobile applications are in need of an Ad-Hoc Wireless Networking Communication Model that provides fast, reliable, and power aware features. Applications such as mobile robots playing soccer, an expedition of robots moving in a hostile area, or a team of rescue robots exploring a building after a disaster are just a few examples that need the above technology. The following sections in this chapter give a brief explanation about Multi-Robot Cooperative Architecture and the popular wireless network standard, namely IEEE 802.11.

1.1 Multi-Robot Cooperative Architecture

A three-level functional architecture is proposed in [1] and [2] for a team of mobile and autonomous robots which are capable of carrying out cooperative tasks. Relationships among robots of the team are modeled using the joint intentions framework. The multi-robot cooperative approach is applied to a Robotic Soccer environment. The cooperative architecture is a general framework for implementing distributed artificial intelligence and intelligent control by using the concept of behaviors. Robot tasks are composed of subsumptive behaviors.

The organizational level establishes the current team strategy based on (1) the team state and (2) the world state. Complexity is reduced by the decomposition of team strategies into individual behaviors, which in turn are composed of primitive tasks. Each strategy is a set of tactics. A running tactic represents an agent behavior that is assigned to a given robot at a given time. Examples of agents are the attacker, goalie, supporter, defender, etc. The team state corresponds to the current set of behaviors under execution. The world or game state consists of (1) game situation and (2) evaluation of situation. The game situation describes the current game mode like kickoff, end-of-game, penalty-for, penalty-against, etc. The team evaluation of current game modes are like (1) losing and close to the end of the game, (2) ball close to our goal, etc. The world-game state refers to what it has been achieved since the beginning of the game and how this may influence the selection of a tactic. Moreover, behaviors are assigned to the individual robots, after a selection from within behavior sets representative of alternative tactics for the strategy selected by the organizational level.

The organizational level determines a strategy and a specific tactic to implement the strategy. The strategy must specify not only the goal to be attained (e.g. attack, defense) but also criteria to check how close to the goal the team is. The tactic consists of behavior sets, whose elements are the behaviors assigned to each individual robot of the team. A tactic is chosen based on the current world state, but also on each agent's current internal state.

In the relational level, the robots define their relationships by negotiating and eventually come to an agreement about some team and/or individual goal. Behavior assignments may also be temporarily modified as a result of inter-robot negotiations. The joint intentions framework provides a foundation for teamwork modeling at the relational level of the architecture.

The individual level handles all the available robot behaviors which form the agent body such as search-ball, walk-to-ball, stand-behind-ball, kick, etc. A behavior corresponds to a set of goal-oriented primitive tasks which are sequentially and/or concurrently executed. A primitive task is a sense-think-act loop. The goal is to accomplish some objective like moving to a pose which includes a robot position and orientation. The sensing data is required to measure to progress in accomplishing the goal like the distance to an object.

At the robot architecture level, each individual robot is provided with all the three levels of the team functional architecture. However, the organizational level is only active in the current head robot and the other robots have disabled organization layer to provide some

fault-tolerance. When the head has fault like loss of power or other, a new head is selected.

At the state machine level, the strategy is determined at the organizational level by a state-machine whose transitions are traversed upon the matching of specific world states, and whose states define the current strategy. Therefore, strategies change when the world state, as perceived by the team, changes. The tactic selection, including behavior selection, negotiation, and temporary behaviors modification, is implemented by relational rules at the relational level.

In [3], the formulation is based on the Joint Commitment Theory for which the commitments among team mates are established in the relational behaviors. The above three layers are processed sequentially from the selection of a role, a commitment, and an individual behavior based on the robot's role and commitment. In the pass relational behavior, two robots set up a long term commitment, in which several individual behaviors are executed. The pass relational behavior is based on the synchronization of both players' actions, which is achieved by communication, and the execution of their individual skills. One of the robots is referred to as the kicker; he starts having the ball and will try to kick the ball in the direction of the other robot, the receiver, who has to intercept the ball.

The Joint Commitment Theory is used to select relational behaviors. Predefined logical conditions can establish a commitment between two agents. Once a robot is committed to a relational behavior, it will pursue this task until one or more conditions become false, or until the goal has been accomplished.

One of the agents, who set a request for a relational behavior, takes the initiative for the relational behavior. A potential partner checks if the conditions to accept are valid. If so, the commitment is established. During the execution of the commitment, the changing environment can lead to failure or success at any time, in which case the commitment will be ended. In general, a commitment consists of three phases: Setup, Loop and End. The setup and ending of a commitment are used for synchronization. Only in the Loop phase, the robots select primitive behaviors concerning the commitment in order to achieve their joint goal.

1.1.1 Auction-Based Joint Commitment

The game status and other logical conditions may trigger the need for a joint commitment, which leads two or more agents to cooperate, as part of a relational behavior, in a given task like the ball pass behavior. In general, a commitment consists of three phases: (1) *Setup,* (2) *Execution Attempt,* and (3) *End or Release,* where *Setup* leads to searching for a potential partner, *Execution Attempt* implements the commitment behavior with proper synchronization and interception, and *End* is to end the commitment or its interruption in the case of a dynamic change in game status.

Figure 1.1 shows the state diagram of the dynamic commitment. The initiator or kicker broadcasts an auction (broadcast *B_Auction_Rq*) announcing the detection of an opportunity for scoring and asking the receivers to make a bid (*Reply_to_Auction*) based on their game conditions like availability in some field area and visibility of the goal. The bids are received and analyzed by the initiator which may return a grant message (*Auction_Select*) to the winner bid, which becomes the partner, and the other bidders become free. To re-evaluate the scene, the new states of the kicker and partner become

Aim, *Pass* and *Standby*, respectively, in which synchronization is done through peer-to-peer messaging (*B_Prepare*) for kicker and (*B_Status*) for partner. The dynamic game conditions may change for both the kicker and/or the partner which may lead to:

- If the positioning of the opponents changes and the kicker finds a way to directly kick the ball it must finish the commitment (*B_Res*tart) which causes the partner to become free,
- The kicker completes the pass of the ball (*B_Pass*) which causes the partner to change its state to Intercept, to attempt intercepting the ball, and dynamically kick the ball (*B_kick*),
- While intercepting the ball, the partner may find no opportunity to kick towards the goal. In this case, it restarts as a new kicker where a new commitment is to be attempted.

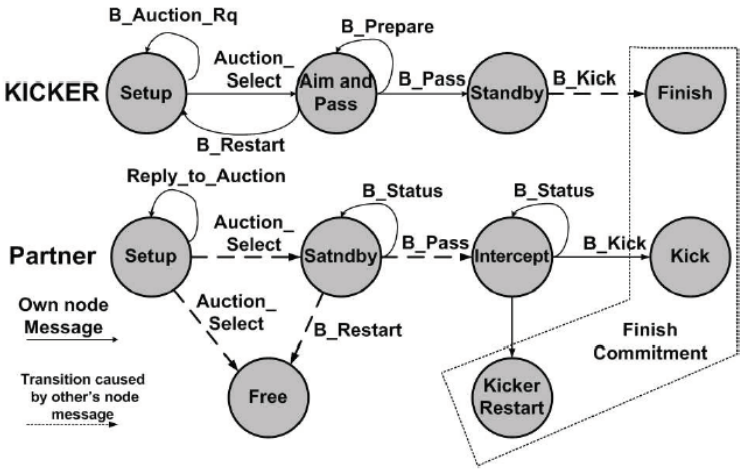

Figure 1.1: Auction-based dynamic commitment using peer-to-peer messaging

The joint commitment is established based on a finite state machine and a messaging system to: (1) synchronize the pass behavior, (2) reiterate the process and extend the pass

to another partner, or (3) break the commitment and search for a new partner depending on dynamic game conditions. To provide dynamic joint-commitment and needed synchronization a fast, reliable, and power aware communication model is needed for Ad-Hoc wireless networks forming a cooperating multi-robot system. Applications such as mobile robots playing soccer, an expedition of robots moving in a hostile area, or a team of rescue robots exploring a building after a disaster are just a few examples that need the above technology. Current techniques are based on client-server and Publish/Subscribe which are not suitable for dynamic joint-commitment. There is need for reliable peer-to-peer model to handle the unpredictable need for communication in mobile systems.

1.2 The IEEE 802.11 Standard

The most popular and most widely deployed standard for WLANs today is the IEEE 802.11 standard, due to its simplicity, flexibility and cost effectiveness [4]. The IEEE 802.11 standard can be implemented on a chip, and WLANs based on IEEE 802.11 can be deployed easily, as no special setup is needed. The cost of IEEE 802.11 WLANs has dropped for both access points (APs) as well as for interface cards, and most new laptops presently have built in WLAN IEEE 802.11 b/g functionality.

IEEE 802.11 has two modes of operation [5]: (1) Infrastructured mode, and (2) Infrastructure- less mode (Ad-hoc mode).

Infrastructure mode

In infrastructure mode Figure 1.2, the wireless network consists of at least one access point connected to the wired network infrastructure called the Distributed System (DS),

and a set of wireless end stations. This kind of configuration is called a Basic Service Set (BSS). A collection of two or more BSSs forming a single sub-network is called an Extended Service Set (ESS). Since most corporate WLANs require access to the wired LAN for services such as file servers, printers, Internet links, they will operate in infrastructure mode.

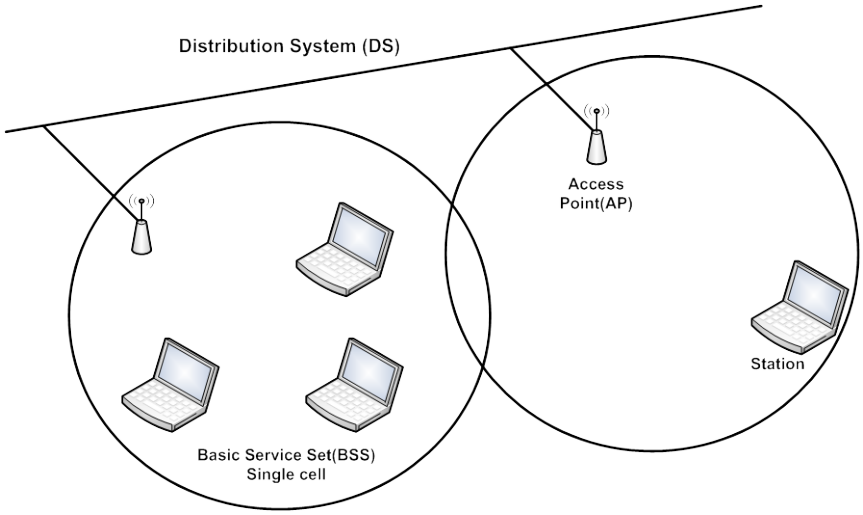

Figure 1.2: Infrastructure mode

Infrastructure-less mode

Infrastructure-less or Ad-hoc mode (also called peer-to-peer mode or an Independent Basic Service Set, or IBSS), is a collection of two or more devices or stations with wireless communication and networking capability, that allows these devices to communicate with each other without the aid of any centralized administrator. They can be either single-hop or multi-hop networks [6]. A single-hop network is one in which all nodes are within the transmission range of each other and any node can reach another node in the network in one hop. In a multi-hop Ad-hoc network, each node acts like a

router to the other nodes in its neighborhood and a node might reach another node in the network only after traversing multiple hops. Figure 1.3 below shows three nodes forming an Ad-hoc network [7].

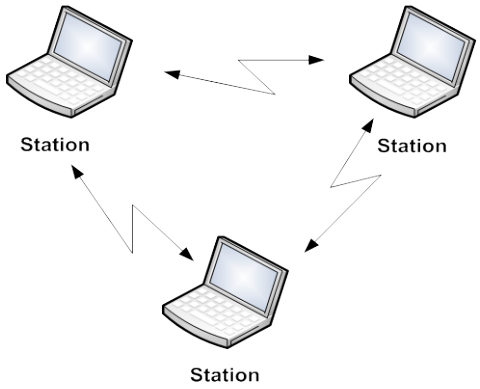

Figure 1.3: An Ad-hoc network example

In an Ad-hoc network, if a sender sends data to a particular single receiver/station, then it is called *unicasting* of data, and if a sender sends data to a selected group of receivers/stations then it is called *multicasting* of data, and if a sender sends data to all the receivers/stations in the network then it is called *broadcasting* of data [8].

1.3 IEEE 802.11 Physical Layer

Mapping IEEE 802.11 MAC frame unit into a format suitable for sending and receiving messages via a wireless medium is done through the IEEE 802.11 Physical Layer (PHY) [9]. Sending or receiving of the messages between two or more stations can be done using one of three implementations: Frequency Hopping Spread Spectrum (FHSS), Direct Sequence Spread Spectrum (DSSS), or Infrared (IR). The FHSS utilizes the 2.4 GHz Industrial, Scientific, and Medical (ISM) band (2.4000-2.4835 GHz), and this band is divided into frequency channels of 1 MHz bandwidth each. In the specification, three

different frequency hopping sequence sets are defined, and each set has 26 hopping sequences. Multiple BSSs coexist in the same geographical area due to different hopping sequences, and this is important to ease congestion and to maximize the total throughput in a single BSS. FHSS uses 2-level Gaussian Frequency Shift Key (GFSK) and 4-level GFSK modulation schemes to specify two access rates 1Mbit/s and 2Mbit/s respectively.

The DSSS implementation uses the 2.4 GHz ISM frequency band, and this band is divided into frequency channels of 11 MHz bandwidth each. The spreading is done with a pre-defined 11-bit chip sequence by chipping each data symbol at 11 MHz in one channel. DSSS Uses Differential Binary Phase Shift Keying (DBPSK) and Differential Quadrature Phase Shift Keying (DQPSK) modulation schemes to specify two access rates of 1Mbit/s and 2Mbit/s respectively.

The IR implementation uses 850 nm to 950 nm wavelengths for signaling. It is designed for indoor use only and requires line of sight or reflected transmission. It uses 16-Pulse Position Modulation (PPM) and 4-PPM modulation schemes to specify two access rates of 1Mbit/s and 2Mbit/s respectively. IEEE 802.11b is a direct extension of the DSSS (Direct-sequence spread spectrum) modulation technique defined in the original standard. Technically, IEEE 802.11b [5] standard uses Complementary code keying (CCK) as its modulation technique. 802.11b has a maximum raw data rate of 11Mbit/s and uses the same CSMA/CA media access method defined in the original standard.

1.4 IEEE 802.11 Medium Access Control (MAC) Layer

The MAC [9] specification defines the way stations access the channel (medium). There are two modes for channel access: Distributed Coordination Function (DCF) and Point

Coordination Function (PCF). DCF is distributed and contention based in nature, and is mandatory in the standard. It can be used with both the infrastructure as well as infrastructure-less configurations. The PCF is a contention-free and centralized access method which is based on polling, and is built on top of DCF. It is optional and can be used only with the infrastructure configurations, as this research is purely based on Ad-hoc networks, PCF does not comes into the picture, hence is not discussed further.

To prioritize the stations to access the medium to transmit data, the IEEE 802.11 standard defines three main interframe spaces. These interframe spaces are: DCF interframe space (DIFS), PCF interframe space (PIFS), and short interframe space (SIFS). The shortest is the SIFS and it is used for acknowledgements (ACKs) and CTS. PCF enabled access points wait for a PIFS duration rather than DIFS. A PIFS duration is less than DIFS and greater than SIFS. The longest, which is the DIFS, is used for regular access in the contention period. If the channel is found busy during the DIFS interval, the station should defer its transmission. The following section gives a brief explanation on how stations use DCF to access the medium.

1.4.1 DCF (CSMA/CA)

DCF [10] is a contention-based access scheme. It is also known as Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) or Listen before talk. In this scheme when a station wants to transmit, it senses the medium; if it is idle for a period of time equal to DIFS, it transmits the frame; otherwise, it defers and waits until the medium becomes idle again for DIFS. In 802.11b, SIFS is 10µs and DIFS is 50µs. It then starts the backoff procedure, where it will chose a uniform random number in the range [0, CW], where CW is the contention window. The backoff interval is calculated as follows:

$$\text{Backoff Time} = \text{Random}() \times \text{aSlotTime}\,[10] \quad (1.1)$$

Where *aSlotTime* is the time length of an empty slot, and is physical-layer dependent.

During the backoff interval, while the medium is sensed idle for a slot time, the backoff counter is decremented by one. If, on the other hand, the medium is sensed busy, the station freezes (suspends) the counter. When the medium becomes idle again for DIFS, the countdown is resumed. When the counter reaches zero, the station transmits the frame. The backoff procedure is invoked between every two successive transmissions. The CW is assigned a minimum value, CW_{min}, at the beginning of the backoff procedure. Its value will be doubled after each unsuccessful transmission; i.e. $CW_i = CW_{min} * 2^i$, where *i* is the stage of the backoff procedure (the number of successive collisions that occurred during this invocation of the backoff procedure). The CW is increased until the stage m is reached; equally stated, until CW reaches CW_{max}. It remains in this stage until the frame is successfully transmitted or the retransmission (retry) limit is reached. After each successful transmission, the CW is reset to the minimum value, CW_{min}. CW_{min} and CW_{max} are fixed values; they are dependent on physical layer (PHY) implementation. For example, CW_{min} and CW_{max} for DSSS are 31 and 1023 respectively, while their values for FHSS are 15 and 1023 respectively.

DCF defines two modes of operation: basic access and request to send/clear to send (RTS/CTS). In the basic access mode, a user first senses the channel status when ready to transmit a packet. If the channel is found to be busy, the user defers its transmission and continues to sense the channel until it is idle. After the channel is idle for distributed inter frame space (DIFS), the user generates a random back-off time before transmitting. Time

after the DIFS period is slotted. Time slot is defined as, the time needed per any user to detect the transmission of a packet from any other user. The back-off counter is decremented as long as the channel is sensed idle, frozen when the channel is sensed busy, and resumed after the channel is sensed idle again for more than DIFS. The user initiates the transmission when the back-off counter reaches zero. Following the frame reception, the destination station waits for a SIFS period, and then sends a short acknowledgement (ACK) control frame to the source station. The ACK frame gives the source station an indication that the frame was successfully received by the destination station. If the ACK frame is not received within a specified time limit, the source station assumes that the data frame was not correctly received by the destination station. In this case, the source station will schedule a retransmission (if the retry limit is not reached) with the CW doubled as explained above. The working of the basic access scheme is shown in Figure 1.4.

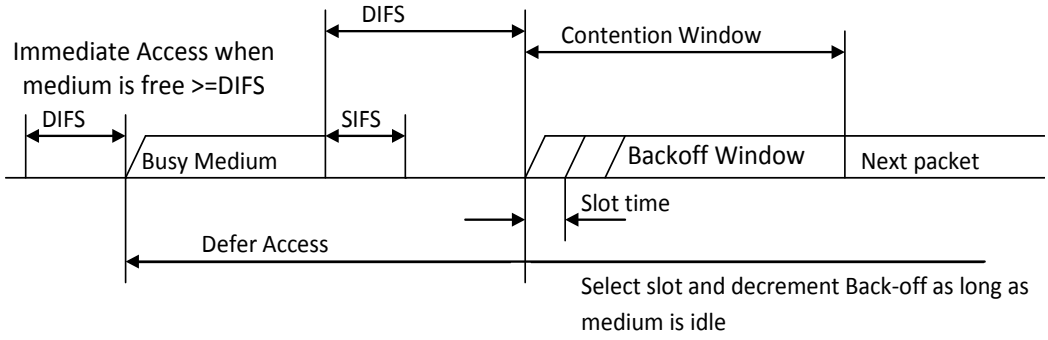

Figure 1.4: Basic access method of DCF

In RTS/CTS access scheme, special control frames support the actual transmission of data frames. The source station first transmits a short RTS frame to the destination station. After a SIFS period following the reception of the RTS frame, the destination

station will transmit another short frame called CTS. If this CTS frame is correctly received by the source station, it will transmit its actual data frame after a SIFS period. Finally, the destination station transmits an ACK frame indicating successful reception of the data frame. Because the RTS/CTS scheme incorporates these four transmissions, it is sometimes called a “four-way handshake”. The advantage of RTS/CTS is twofold. First, the time of collision (if it occurs) is minimized. Second, the hidden terminal problem is solved, where hidden terminal problem is defined as a situation which occurs when there is a station that can hear only one of the communicating stations, and not both. Figure 1.5 shows the RTS/CTS mechanism.

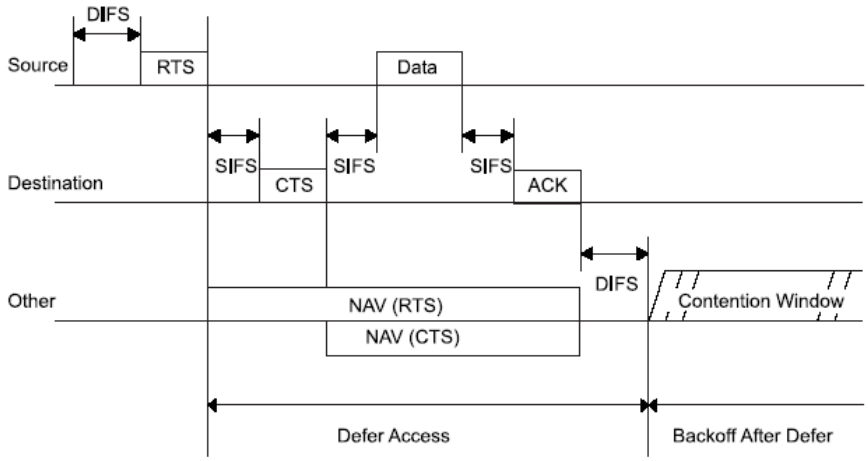

Figure 1.5: RTS/CTS mechanism [10]

1.5 Problem Statement

The IEEE 802.11 Wireless LAN has been deployed increasingly because of the demand for ubiquitous wireless data services. However, the IEEE 802.11 does not support any MAC layer recovery on broadcast frames. The IEEE 802.11b uses RTS/CTS/DATA/ACK scheme for unicasting so as to reduce collisions and to improve

reliability. For broadcasting, IEEE 802.11b simply requires the sender to perform CSMA/CA before broadcasting a DATA frame and no MAC layer retransmission is provided. As a result, due to interference and collisions in the wireless networks, this mechanism leaves broadcasting unreliable. Hence, there is a need to improve the reliability of broadcasting in IEEE 802.11b. In highly dynamic environments like Rescue missions [11] or RoboCup [12], the communications between multiple robots is performed by broadcasting the data to its team mates, also called an *Auction*. In these environments, each robot acts like a peer. There is a need to design an Ad-hoc network that is fast and reliable in terms of communication. Currently available techniques are based on client-server or Publish/Subscribe model, and all communications are carried out using TCP connections because of which the response time is high. We cannot alter the MAC layer of IEEE 802.11b to improve the reliability of the broadcast mechanism it uses, as it is already in commercial use however, we can improve its reliability by applying some schemes or techniques on top of its Application layer. Our problem is to design a Peer to Peer Protocol for Ad-hoc networks that should be reliable and efficient in terms of data broadcasting.

1.6 Challenges

This work involved significant challenges because of the nature of experiments being carried out with real devices. The first challenge was to ensure that the experiments were reproducible. The second challenge was to study the various devices used for the experiment. It took a considerable amount of time to learn how to operate, configure and understand supported features of each device. The features provided by each device were first tested in order to ensure proper functionality. A significant amount of time was spent

to understand the proper configuration. All the experiments were carried out indoors, and it took a considerable amount of time to test our proposed algorithms in this environment. The proposed algorithms were coded in such a way that we do not get any deadlock issue i.e. nodes should not get stuck while communicating with the other peer nodes in the network.

1.7 Organization of Thesis

The rest of the thesis is organized as follows. Chapter 2 gives an extensive literature survey of the related work. Chapter 3 presents the proposed UDP Reliable Broadcast Algorithms. Chapter 4 presents the Experimental Setup and a brief overview of the devices used in the Experiments. Chapter 5 presents the Experimental Results and the Performance Evaluation of the Proposed Algorithms. Chapter 6 concludes the study, and proposes future direction of work.

CHAPTER 2

LITERATURE REVIEW

This chapter reviews the proposed algorithms for Broadcast/Multicast over wireless networks, as found in the literature. It talks about protocol classifications, pros and cons of each scheme, and discusses how they relate to our work.

Much work has been done to improve the reliability of the CSMA/CA Broadcast/Multicast mechanism. The protocols or the schemes found in the literature can be classified into two categories. One group provides reliable Broadcast/Multicast by extending the MAC protocol by altering the RTS/CTS mechanism, and the other group provides reliable Broadcast/Multicast by extending the IEEE 802.11 wireless interface by the use of ACKs without changing the MAC protocol i.e. Reliability is achieved by applying some technique on top of the Application layer.

2.1 Reliable Broadcast/Multicast MAC Protocols by altering the RTS/CTS mechanism of IEEE 802.11

Most of the protocols in this section are ACK-based, but some of them are ACK as well as NACK-based, and these protocols are designed to provide reliable Broadcast/Multicast

by extending the MAC. These protocols basically alter the RTS/CTS mechanism to avoid collisions between RTS/CTS control frames.

Leader Based Protocol (LBP) [13] was designed to provide reliable multicast over WLANs. In this protocol one of the receivers is chosen as a leader and this leader is responsible for supplying CTS in response to the RTS, and an ACK in response to the data packet. This protocol works as follows: a) First in slot1, the base station or the sender sends a multicast RTS to the receivers. b) In slot2 the leader will send CTS upon hearing RTS if it is ready, otherwise does nothing. c) In slot 3, if CTS was heard during slot 2 by the sender then begin multicast transmission of data, or if no CTS was heard during slot 2, then backoff and go to step (a). d) leader of the receivers will send the ACK if the data packet was received without error and NAK if the data packet is received with error, and other receivers will do nothing if the data packet is received without error and sends NAK if the data packet is received with error. That means LBP uses both ACKS and NAKs from receivers as feedback to the sender.

In Delayed Feedback Based Protocol (DBP) [13], a random timer is used to avoid the CTS collisions. The protocol works as follows: a) First the base station or the sender will multicast the RTS and start a timer of timeout period T, and expects to hear CTS before the timer expires. b) on hearing the RTS each receiver starts timers with an initial value selected randomly from 1,2,3,…,L, and decrements the timer by 1 in each slot, if a CTS is heard before the timer expires, the timer is freezed and if the CTS is not heard before the timer expires then CTS is sent. c) If no CTS is heard by the sender within the time period T, then it backs off and goes to step (a), and if CTS is heard by the sender within the time period T, then it starts transmission of the data packet. After the data packet is

transmitted the sender is prepared to transmit the next data packet and go to step (a), step (d) is executed only if there is multicast transmission. d) The receivers will do nothing if they receive the data packet without error, and contend for the channel to send NAK if the packet received is in error. PBP [13] is similar to the DBP except that in PBP the group members send out CTS in the slot following RTS with a certain probability instead of waiting for a number of time slots to send CTS. This probability is chosen based on the number of group members.

The Broadcast Medium Window (BMW) [14] protocol is designed to raise the reliability of broadcast service in IEEE 802.11 and is composed of RTS/CTS/DATA/ACK. In this protocol the sender broadcasts the data reliably by unicasting to each 1-hop neighboring receiver at a distance of 1-hop. The BMW protocol treats broadcasting as a set of multiple unicast operations. For each broadcast, the sender unicasts it to the neighboring nodes using RTS/CTS/DATA/ACK. First, the sender exchanges RTS/CTS with one of the neighboring nodes and after that transmits data to that particular node, and waits for an ACK. After receiving the data, the neighboring node will send an ACK to the sender and this whole process of RTS/CTS/DATA/ACK is repeated for all the remaining nodes. If the neighboring nodes fail to receive the data frame, they send RTS with the sequence number of the missing data frame. The reliability of this scheme is improved at the cost of lowered efficiency.

Batch Mode Multicast MAC Protocol (BMMM) [15] is designed to support the reliable multicast in IEEE 802.11. It is an enhanced version of BMW, the transaction of BMMM between the sender and member nodes is a sequence of multiple RTS/CTS exchanges, data packet transmission, and multiple Request ACK (RAK)/ACK exchanges. During

this sequence, there is no contention-based channel access. Therefore, compared to BMW, BMMM reduces the overhead due to multiple contention periods of the access channel for transmitting RTS/ACK.

In other words, if number of 1 … n received nodes exist, then sender transmits RTS to node 1 and receives CTS, so sender promises to ready to node 1 and exchange RTS/CTS message to 2, 3, ... n node sequentially. After all, senders confirm to all nodes that plan to transmit data and ready. When it finishes RTS/CTS message exchange, sender transmits data to all nodes and exchange RAK/ACK to all nodes that certified result of data transmission for all received nodes. However, there is still overhead of multiple control packets of RTS, CTS, RAK, and ACK. This overhead increases as the number of nodes increases.

In the BACK (Backoff Acknowledgment) window scheme [16], which is defined as the time interval between the end of the SIFS and the end of the DIFS after a DATA frame is introduced, the BACK window is divided into minislots, and each receiver randomly selects one of the minislots to transmit an acknowledgment. If the number of received BACKs of the sender is less than the number of its active receivers, the sender needs to rebroadcast the frame. Hence, as long as one BACK is not received (due to the collision of BACKs, or the collision of a BACK and another frame, or some receivers moving away from the senders transmission range), this requirement is not met and retransmission is performed unnecessarily, even though the DATA has been received by all the receivers.

Extended Implicit Acknowledgment (EIA) [17] protocol has been designed to provide both reliable as well as an efficient Multicast mechanism for WLANs. To improve the efficiency, the control overhead is reduced by using an implicit acknowledgment scheme. The main idea to reduce the control overhead is that if the sender has at least two Multicast packets to be transmitted to a group of receivers, the sender requests the receivers not to transmit an explicit ACK. When RTS/CTS for a second data packet is initiated, the receiver acknowledges the receipt of the first packet by piggybacking the ACK of the first data packet with the CTS. Similarly the CTS frame of the third packet carries the ACK of the second data packet, and soon. When the last data packet is to be multicast, RTS is used to explicitly notify the receiver group members to send an explicit ACK packet. To improve the reliability, collisions among control frames are avoided. In order to avoid collisions among CTS frames of a multicast group, each group member is assigned a certain priority. The CTS frame is sent one after another based on priority, so that the collision of CTS frames is avoided. Each receiver calculates the waiting time based on its priority and sends a CTS signal when the timer expires.

In the Robust Broadcast scheme [18] retransmissions are performed when there is a collision detected for a broadcast transmission. It employs the RTS/CTS/DATA mechanism. In this scheme one of the neighboring nodes called the collision detector is selected for RTS/CTS handshake. When data is broadcast after an RTS/CTS exchange with the collision detector, an ACK is sent back to the sender. The collision detector is responsible for sending back an ACK when data is received in each broadcast. First, the sender transmits an RTS addressed to the collision detector. On hearing the RTS the collision detector responds to the sender by sending back the corresponding CTS frame.

If no CTS is received by the sender, it performs a backoff and retries later. After the RTS/CTS exchange, the sender broadcasts the data and waits for the ACK from the collision detector. The collision detector sends an ACK to the sender after receiving the data packet. For optimization of the scheme, the last packet is sent without RTS/CTS. The choice of collision detector will be a base station in centralized wireless networks, and in Ad-hoc wireless networks, the collision detector will be the source of the last message sent over the medium. The problem with this mechanism is that the RTS/CTS exchange can only avoid collisions on the collision detector but not for other neighbors. Hence, avoidance of hidden terminal problem in the sender's neighborhood is quite limited.

2.2 Reliable Broadcast/Multicast protocols for IEEE 802.11 by applying some techniques on top of Application layer

The protocols in this section provide reliable Broadcast/Multicast by extending the IEEE 802.11 wireless interface by the use of ACKs without changing the MAC protocol. Reliability for Broadcast/Multicast is achieved by requesting an ACK from the receivers by each sender.

In [3], Al-Mouhamed et al. implemented and evaluated an auction based communication model using (1) TCP Peer to Peer Scheme, (2) UDP Peer to Peer (UPTP) Scheme, and (3) UDP Broadcast and Token Passing (UBTP) scheme. The performance evaluation reports that the peer to peer communication using UDP broadcast and token passing scheme performs better than the other models used. In the auction based communication, auction is always requested from the head node, and other nodes always reply to the

auction or perform synchronization related tasks if required by the auction scheme. The schemes presented in this paper are based on UDP and TCP communication protocols, to accomplish the auction.

In the TCP Peer to Peer (TPTP) Scheme, the head node communicates with each node that is included in the auction using TCP packets. All the requests and replies of the auction are performed over the TCP communication link. The TPTP scheme is reliable but is not scalable as the head node has to open a new TCP connection for each node.

In the UDP Peer to Peer (UPTP) Scheme, the head node communicates with each node that is included in the auction using UDP packets. The UDP packet contains the IP address of the destination node, and is sent to the node whose IP address is there in the UDP packet. The head node contacts the first node using a UDP packet, and if it gets the response from the first node through a UDP reply then it contacts the next node for the auction. If any of the nodes does not reply, then the head node retries N times and then moves to the next node. This scheme is reliable if the value of N is kept large enough ($N > 10$), but is not scalable.

In UDP Broadcast and Token Passing (UBTP) Scheme the head node broadcasts a UDP packet that contains an order of the neighboring nodes, which should be adhered to for replies. The first node that is in the sequence replies to the head node and at the same time sends a token to the second node in the sequence. Similarly, the second node replies to the head node and transfers the token to the third node in the sequence, and soon. If any of the nodes does not receive a token, it will reply to the head node after a time out of T milliseconds.

It is shown that TCP communication has the highest delay in auctioning, and UBTP has the least delay. TPTP based communication has large scattered time overhead and lacks scalability. The stability and responsiveness is improved by using UPTP. However, the shortest and most stable times were obtained for UBTP. The main drawback of the UBTP and UPTP schemes is that are they are not purely peer to peer in nature. They involve a head node for initiating the communication and other nodes simply reply back to the head node.

The Round Robin Acknowledge and Retransmit (RRAR) [19] protocol improves the reliability of IEEE 802.11 Broadcast mechanism. In this protocol, when a data frame is ready for transmission, after finishing the collision avoidance phase, the sender broadcasts the data frame. The header of the data frame contains the address of one of the neighboring node which is responsible for sending an acknowledgment called Broadcast Acknowledgment (BrAck). The BrAck contains the sequence number ‘S’ of the latest data frame received and a bitmap specifying the previous data frames with reference to ‘S’ that are lost or received. For each new data frame, the sender selects a different neighbor to reply back with a BrAck in round robin fashion. The DATA/BrAck is done for all the neighbors in round robin fashion, and then the sender checks the bitmap of each neighbor in the received BrAck and retransmits those data frames which are missing or lost to the specified neighbor if they are still stored in the memory.

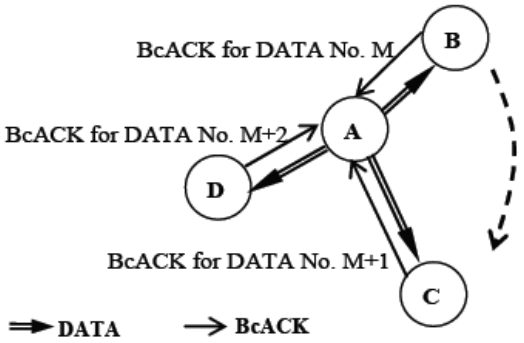

Figure 2.1: Round Robin Acknowledgment and Retransmit [19]

As we can see from Figure 2.1 above, the sender A first broadcasts the data frame number M with its header of it containing the address of its neighbor B. The node B then sends back BrAck to A specifying the data frame sequence number received, and the bitmap. Subsequently, sender A selects other neighbors such as C and D to send the data frames M+1 and M+2 to, in round robin fashion. Upon receiving the data frames, nodes C and D send back an acknowledgment BrAck to the sender A.

The Random Peer to Peer communication (RP2P) [20] protocol is implemented on a group of 10 robots using TCP connections and 802.11b wireless network interfaces. The author is not concerned with what actually is transmitted, but rather how it gets transmitted. In RP2P, each robot listens for messages from its teammates for a period of time known as the communication round. At the end of each communication round, the program sends a message to a randomly selected teammate. These messages are directly sent from sender to receiver in a single hop. For every communication round, each robot randomly selects a teammate and sends it a message, and this is all carried out asynchronously. Messages are sent continuously in a proper functioning RP2P system.

In this scheme robots first carry out an initialization and then wait until they receive a "start" command broadcast over a UDP port from a central controller. Once the "start" command is received, the robots record their local time and keep this recorded time for the remainder of the trial. All peer-to peer communication is carried out via TCP, which guarantees that all messages will arrive at their destination without error. The robots are provided with a time to use as τcom for the trial at run-time. For every τcom ± 15% seconds, a robot will randomly select a teammate and send it a message. For every message that a robot sends or receives, it logs the time at which it was sent/received along with the IP of the sender/recipient. A trial ends when a "stop" command is received over the UDP port. The main drawback of this technique is that all the peer to peer communications are carried out using TCP, which guarantees that all the messages will arrive at their destinations at the cost of increased time delays.

None of the above techniques is purely peer-to-peer based. Most of the schemes are Client/Server based or centralized control based. Hence, there is a need for an Ad-hoc Wireless Networking peer to peer Communication Model using IEEE 802.11 wireless interface that provides fast, reliable, and power aware features for the effective implementation of dynamic cooperative architectures in autonomous robotics. Since we cannot alter the MAC layer of IEEE 802.11b to improve the reliability of broadcast mechanism as it has already been implemented, we can improve the reliability by applying some schemes in the Application layer.

CHAPTER 3

THE UDP P2P RELIABLE PROTOCOL

3.1 Introduction

The IEEE 802.11 Wireless LAN has been deployed increasingly because of the demand for ubiquitous wireless data services. However, the IEEE 802.11 does not support any MAC layer recovery on Broadcast frames. Besides, in real channel conditions, an unsuccessful transmission may happen due to a link error or collision. Hence there is a need for an Ad-hoc Wireless Networking Communication Model using the IEEE 802.11 wireless interface that provides fast, reliable, and power aware features for the effective implementation of dynamic cooperative architectures in autonomous robotics. Specifically, a fast and reliable collective communication is needed to implement real time joint-commitment in highly dynamic environments, Rescue missions and RoboCup.

Wireless networks can be developed using different networking models. Following are the three most common network models.

1) Client/Server Model [21] [22]: In this network model, one node in the network is assigned as a server and other nodes as clients. The server generally performs the majority of the processing tasks. The clients initiate a connection with the server when

they want to communicate with the server. This model is not suitable for Auction based networks because: (1) In the network all nodes have equal computational power, therefore heavy computational load on one computer causes delays in the whole network, and (2) In the applications of auction schemes, any node can initiate communication (auction) with any number of the remaining nodes, so each node should have both client and server capabilities at the same time.

2) Publish/Subscribe Model [23] [24]: This model consists of publishers and subscribers. The publisher is unaware of the recipients of its messages and rather it publishes messages to a class of subscribers. The subscribers can receive messages from the classes in which they are interested without any knowledge about the publisher. In this way the publisher and subscriber are decoupled in this model. While this model looks suitable for the auction schemes, it has many extra features that are not needed by the auction schemes. First and foremost problem is that the target nodes for each auction are known to the initiator, and once an auction is done the list of nodes which participated in the last auction become unimportant for other nodes, because successive auctions cannot always be interrelated. Secondly, the target nodes for any auction are determined dynamically by the initiator based on the application requirements. Therefore, classes in this model need to be changed dynamically for each auction or too many classes need to be formed to meet requirements of each auction. That is an unnecessary burden for short communications like those used in the auction schemes.

3) Peer/Peer Model [25] [26]: In this network model, each node can connect to any other node or group of nodes and send/receive the data. The connections are Ad-hoc and they last until the initiators or any other nodes want to terminate the connection. Features can

be added into the basic Peer/Peer model, to add functionality needed by the application. Peer/Peer models can use both TCP and UDP protocols. The following features of the auction schemes are best implemented in this network model: (1) Communication among the nodes in the auction is very short. (2) The behavior of each node is controlled by the node itself so there is no need for continuous data transfer. (3) Many features like broadcast or multicast can be used in Peer/Peer networking. (4) Any node can initiate communication with any other node or group of nodes using multicasting.

The currently available techniques in the literature are based on Client/Server and Publish/Subscribe communication model, which are not suitable for our problem.

In this chapter we presented the proposed solutions to solve the problem of reliable broadcast over IEEE 802.11b environments by employing the peer to peer communication model. The basic idea to form an Ad-hoc wireless network using IEEE 802.11b wireless network interfaces is that it is inexpensive, readily available and widely supported for the implementation of dynamic cooperative architectures in autonomous robotics.

In the following Sections, we discuss our proposed schemes. In Section 3.2 we discuss the UDP P2P Reliable Broadcast with Token Passing scheme and in Section 3.3 we discuss the UDP P2P Reliable Distributed Broadcast scheme.

3.2 Protocol 1: UDP P2P Reliable Broadcast with Token Passing scheme

3.2.1 Description

In UDP P2P Reliable Broadcast with Token Passing (P2P-RUBTP) scheme each node acts like a peer and a symmetric code is run on all the peers. Any one of the peers broadcasts UDP packets to all other peer nodes in the network, and the order in which the nodes must respond or reply to the sender. The UDP packet contains the DATA and the order of sequence of the other peers in which they should reply. Each peer node replies to the sender peer node which initiated the broadcast by sending an ACK, and passes a token to the next peer node which is next in line in order. If the next node receives the token, then it will reply to the sender with ACK and passes a token to the next node in order. If the next node does not receive the token from its neighboring, it will reply with an ACK to the sender peer which had initiated the Auction after a time out period of T milliseconds. If the next node receives a token after a time out of T milliseconds it will just neglect the token. The last node in the sequence will reply only with an ACK to the sender in each broadcast. If the peer initiator node does not receive ACK from any one of the nodes it will do a second broadcast to the nodes that did not send an ACK. The sender peer will conclude the broadcast if it receives ACK from all the other nodes, otherwise it will broadcasts to the selected nodes that have not replied with ACK until they reply, or the initiator gives up. The operation of the UDP P2P Reliable Broadcast with Token Passing scheme is shown in Figure 3.1.

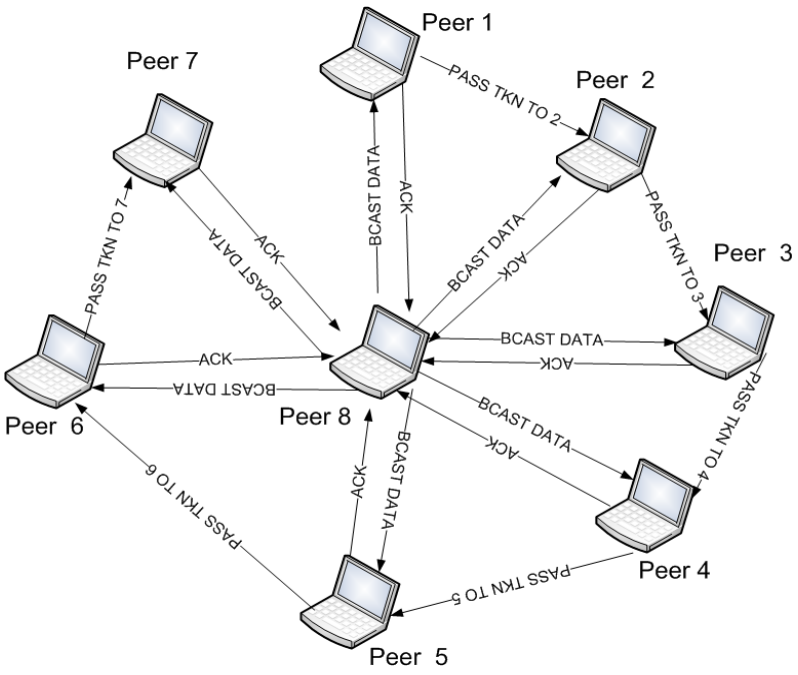

Figure 3.1: UDP P2P Reliable Broadcast with Token Passing scheme

In Figure 3.1, Peer node 8 broadcasts the UDP packet containing data and the order of sequence in which all other peer nodes should reply. Suppose the order of sequence is 1, 3, 7, 4, 6, 5, and 2, then after receiving a UDP packet, the first peer i.e., node 1 will reply to node 8 by passing an ACK, and will pass a token to node 3. After receiving the token, node 3 will reply with an ACK to node 8. If node 3 does not receive a token, it will reply with an ACK to the peer node 8 after a time out period of T milliseconds. All the nodes reply according to their order in the sequence as explained above. The last node i.e. peer node 2 will only send an ACK to peer node 8 as there is no node left in the sequence to send a token to. Figure 3.2 shows the flow chart of the proposed algorithm UDP P2P Reliable Broadcast with Token Passing scheme.

3.2.2 Flowchart

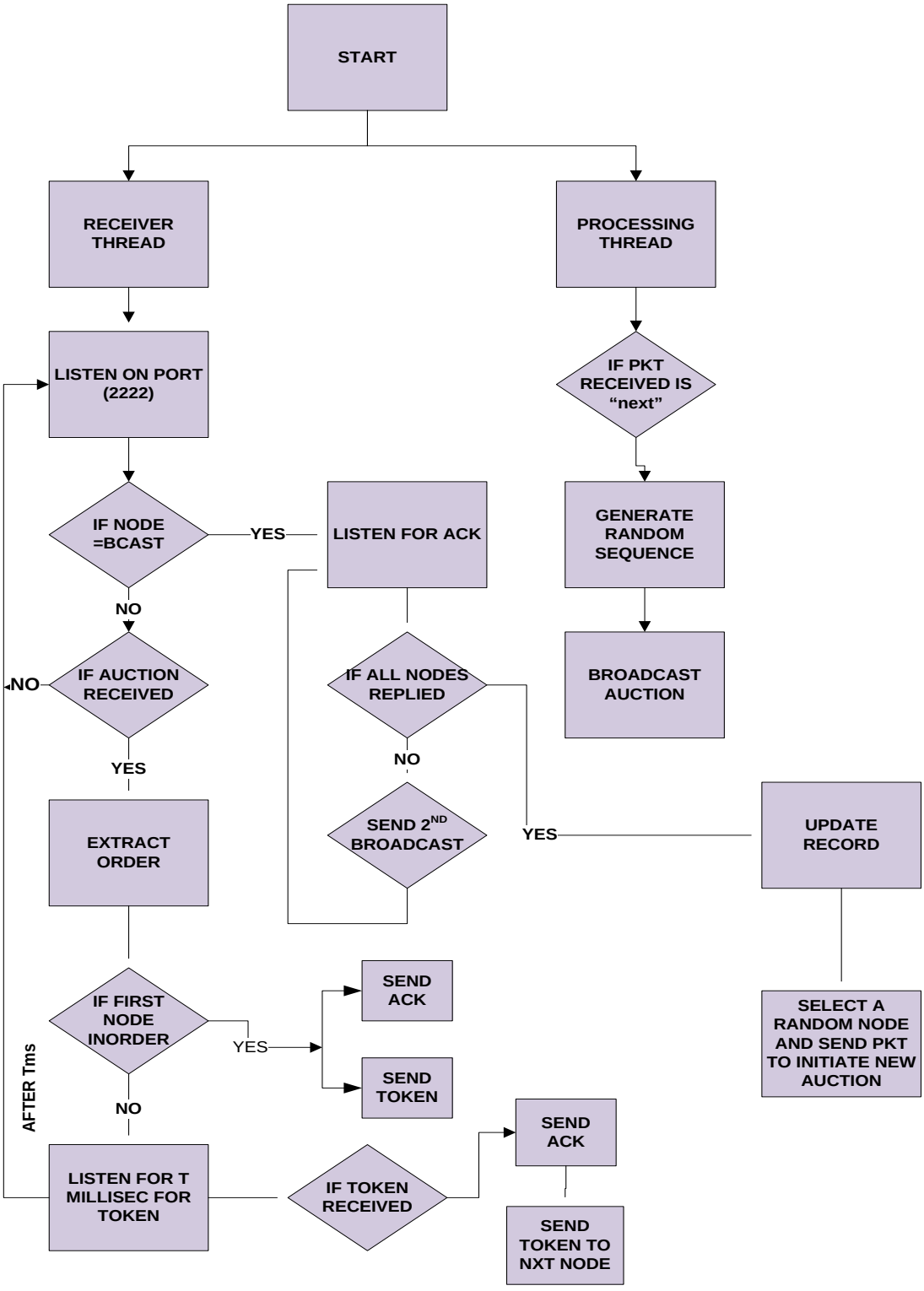

Figure 3.2: Flow chart of UDP P2P Reliable Broadcast with Token Passing scheme

The main parts of the algorithm are:

1. Main Module (Start Module)
2. Receiver Thread (RT)
3. Processing Thread (PT)
4. Sender Module

The Main Module initializes the program and creates two Threads, Receiver Thread and Processing Thread. The function of Receiver Thread and Processing Thread is shown in the above flow chart (Figure 3.2). The following sections give the description of each module in detail.

3.2.3 Receiver Thread

The Receiver Thread performs the following steps:

- Listen on the Port: When the Main Module initializes Receiver Thread, it starts listening on the port 2222 for the UDP packet; once the packet is received it goes to the next step.
- Extract the Order: Once the packet is received, Receiver Thread extracts the IP address of the sender node from the received UDP packet, then it will perform the following tasks:
    - If the sender node is itself i.e., auctioning node, then it goes to the listening mode, and listens for Acknowledgments from all other nodes. If all nodes replied with ACKs then it updates the record i.e. writes the time taken to complete an auction into the file. If any one of the nodes does not reply within the specified time, then it broadcasts the same packet a second time to the

specified nodes that did not reply with an ACK. The sender node will conclude the broadcast if it receives ACK from all the other nodes, otherwise it will broadcast to the selected nodes that have not replied with ACK until all nodes reply.

- If the node receives an auction i.e., it is not an auctioning node, and then extract the order from the packet in which the node must send an ACK to the sender node. If it is the first node in order, then immediately send an ACK to the auctioning node and send a token to the next node which is next in line in the order. If it is not the first node in order, then listen for the token for Tms. If token is received within Tms, then send an ACK to the auctioning node and send a token to the next node which is next to it in order. If token is not received within Tms i.e. after Tms send ACK to the auctioning node and send a token to the next node which is next to it in order.

- Next Auction: After all nodes replied with ACK then send an initiation packet to a randomly selected node through the Sender Module.

3.2.4 Processing Thread

The Processing Thread performs the following steps:

- Generate Auction: The Process Thread generates a random sequence or the order in which all nodes should reply and added it to the UDP packet and broadcast (Auction) it through Sender Module.
- Update Record Method: This method maintains a file related to the completion times of the Auctions. Whenever all the nodes replied with ACK's, the time taken to

complete an auction is recorded in the file i.e. the time from the start of the Auction to the time when all nodes replied with ACK's.

- Process the Auction Request: Whenever the Receiver Thread receives an initiation packet by selecting a random node; the Process Thread is restarted again.

3.2.5 Sender Module

Sender Module performs the following tasks:

- Sends the Auction or the initiation packet: The main purpose of this method is to send an auction started by the Process Thread and it is also used to send an initiation packet to the randomly selected node.
- Sends ACK: This method is also invoked whenever a node wants to send an ACK to the auctioning node.
- Mostly Inactive: Most of the time this method is inactive as this method is invoked only when there is something to be sent.

3.3 Protocol 2: UDP P2P Reliable Distributed Broadcast scheme

3.3.1 Description

In the UDP P2P Reliable Distributed Broadcast scheme, each node acts like a peer, and a symmetric code is run on all the peers. Any one of the peer broadcasts UDP packets to all other peer nodes in the network, and the order in which the nodes must respond or reply to the sender. The UDP packet contains the DATA and the order of sequence of the other peers in which they should reply. Each peer node replies to the sender peer initiator node, by sending ACK. The ACK frame is sent one after another based on their position in the

order of sequence of the receivers. Each receiver node calculates the waiting time based on its priority (order in the sequence) and sends an ACK when the timer expires. Each node waits for time Tms (position of it in the order of the sequence multiplied with the T acknowledgment time (ms)). If the peer node which started the auction does not receive an ACK from any one of the node, it will do a second broadcast to the nodes that have not sent the ACK. The sender peer will conclude the broadcast if it receives ACK from all other nodes, otherwise it will broadcast to the selected nodes that have not replied with an ACK until they reply or the originator gives up.

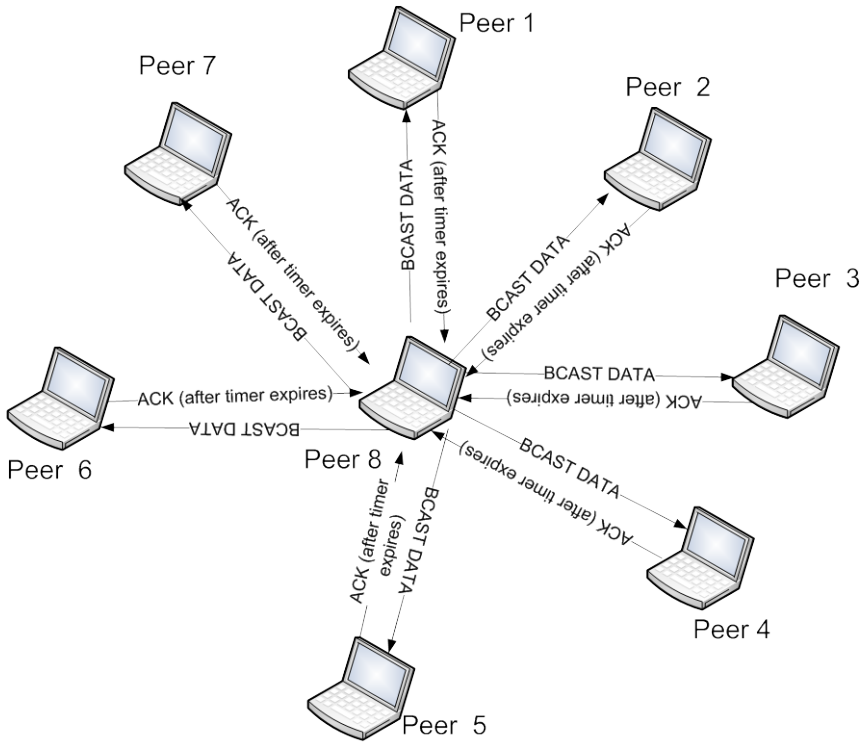

Figure 3.3: UDP P2P Reliable Distributed Broadcast scheme

In Figure 3.3 above peer node 8 broadcasts the UDP packet containing data and the order of sequence in which all other peers should reply. Suppose the order of sequence is 1, 3, 7, 4, 6, 5, and 2, then after receiving the UDP packet. The first peer node 1 will reply to

the node 8 by passing ACK and remaining nodes will start timers based on their positions in the order of sequence. The remaining nodes will reply with ACK to the peer node 8 after their timers expire. All the nodes reply according to their order in the sequence. Figure 3.4 below shows the flow chart of the proposed UDP P2P Reliable Distributed Broadcast scheme.

3.3.2 Flowchart

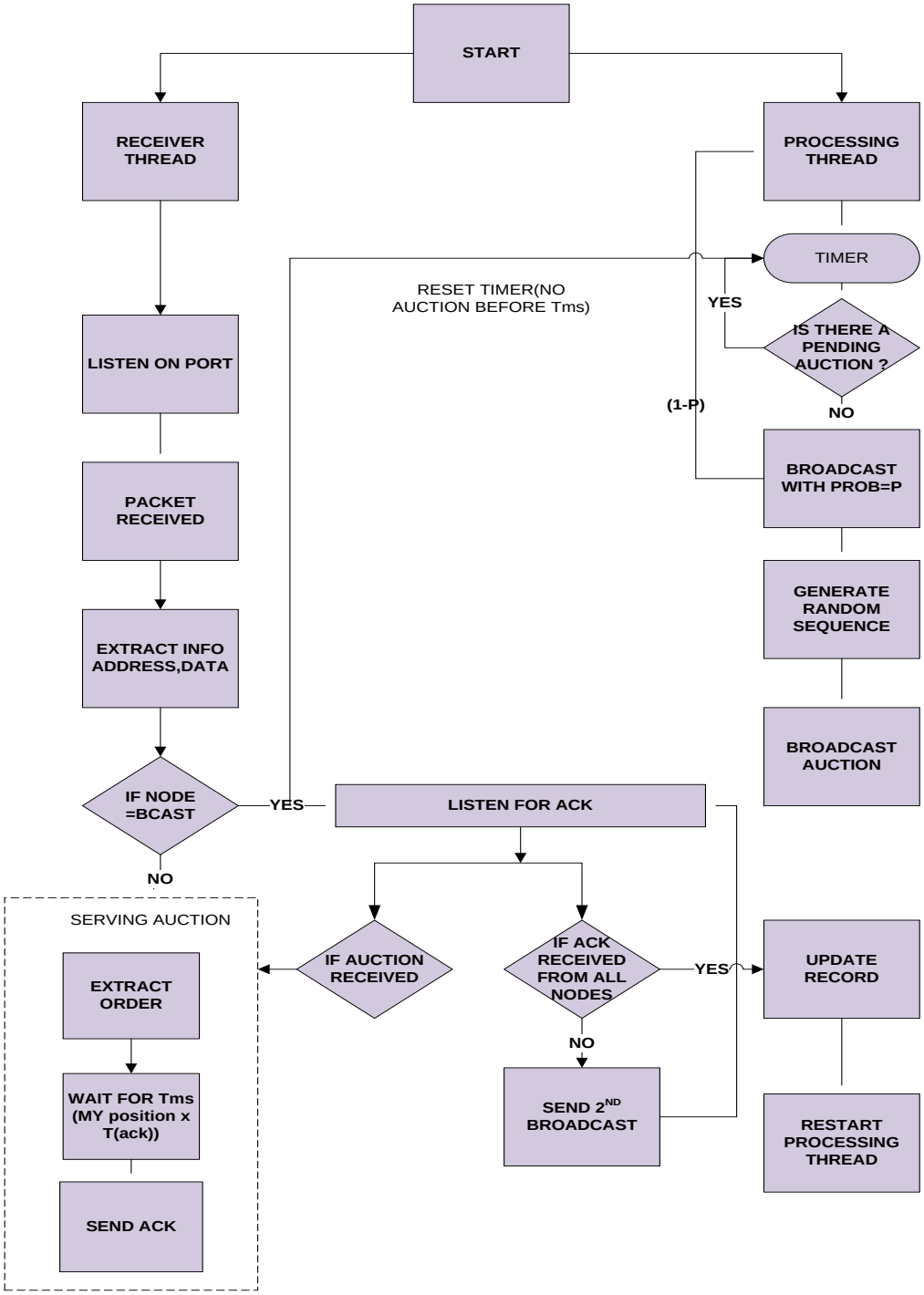

Figure 3.4: Flow chart of UDP P2P Reliable Distributed Broadcast scheme

The main parts of the algorithm are:

1. Main Module (Start Module)
2. Receiver Thread (RT)
3. Processing Thread (PT)
4. Serving Auction Module
5. Sender Module

The Main Module initializes the program and creates two Threads, Receiver Thread and Processing Thread. The function of the Receiver Thread and the Processing Thread is shown in the above flow chart (Figure 3.4). The following sections give the description of each module in detail.

3.3.3 Receiver Thread

The Receiver Thread performs the following steps:

- Listen on the Port: When the Main Module initializes the Receiver Thread, it starts listening on the port for the UDP packet; once the packet is received it goes to the next step.
- Extract the Info (Address, Data): Once the packet is received, the Receiver Thread extracts the IP address of the sender node from the received UDP packet, then it will do the following tasks
    - If the sender node is itself i.e., auctioning node, then go to the listening mode and listen for Acknowledgments from all remaining nodes. Here we may have two cases (i) Again it may receive auction from other nodes, if it receives auction it replies back with ACK. (ii) It may receive ACK from other nodes,

and if all nodes replied with ACKs then update the record i.e. write the time taken to complete an auction into the file. If any one of the node doesn't reply within the specified time then broadcast the same packet a second time to the specified nodes that did not reply with an ACK. The sender node will conclude the broadcast if it receives ACKs from all other nodes, otherwise it will broadcast to the selected nodes that have not replied with an ACK until all nodes reply.

- If the node receives an auction i.e., not an auctioning node then go to Serving Auction Module and restart the timer (no auction before T ms). Extract the order from the packet in which the node must send an ACK to the sender node, if it is the first node in order then immediately send an ACK to the auctioning node. If it is not the first node in order then calculate the waiting time based on the order and send an ACK to the sender when the timer expires.

3.3.4 Processing Thread

The Processing Thread performs the following steps:

- Starts Timer and Check for Pending Auctions: As soon as the processing thread is started, it starts a timer of T ms and after the timer expires it checks for any pending auctions, if there is any pending auction it resets the timer, and this continues until there is no pending auction, and after that it goes to the next step.
- Generate Auction: Each node is made to broadcast with a certain probability 'P'; if it gets the probability then it goes to the next step. If it does not get the probability i.e.,

if it gets (1-P) then the Processing Thread is restarted again. After getting the probability ‘P’, the Processing Thread generates a random sequence or the order in which all nodes should reply and adds it to the UDP packet and broadcast (Auction) it through the Sender Module.

- Update Record Method: This method maintains a file related to the completion times of the Auctions. Whenever all the nodes replied with ACK’s, the time taken to complete an auction is recorded in the file i.e. the time from the start of the Auction to the time when all nodes replied with ACK’s.

3.3.5 Serving Auction Module

Whenever an auction is received this module is invoked. Following are the steps performed by this module:

- Extract order: It first extracts the order from the UDP packet and then goes to the next step.
- Calculate waiting time: After extracting the order, it calculates the waiting time T ms based on the nodes order in the sequence. The waiting time is calculated as

  Waiting time = Node_Position x T (Ack), where T (Ack) is the acknowledgment time.
- Sends ACK: After the waiting time expires, each receiver sends an ACK to the sender node through the Sender Module.

3.3.6 Sender Module

Sender Module performs the following tasks:

- Sends the Auction packet: The main purpose of this method is to send an auction started by Processing Thread.
- Sends ACK: This method is also invoked whenever a node wants to send an ACK to the auctioning node.
- Mostly Inactive: Most of the time this method is inactive as this method is invoked only when there is something to be sent.

CHAPTER 4

EXPERIMENTAL SETUP

4.1 Introduction

In this chapter, we discuss the experimental setup and the hardware devices used for the experiments. The hardware consists of Stargate boards [27] equipped with wireless networking cards. The Stargate board is a powerful single board computer that consists of an Intel 32-bit, 400 MHz Xscale processor and 96 MB of memory (SDRAM and Flash). The Stargate also has a daughter board that contains a socket for the wireless card and Ethernet interface. The software of the Stargate comprises of Linux OS with drivers for all peripherals and Java Runtime Environment (JRE). The Stargate system directly supports applications around Intel's Open-Source Robotics initiative, as well as Tiny OS based Wireless Sensor Networks, and the Smart Dust Technology.

In our experiments each Stargate has an Ambicom IEEE 802.11b wireless card. The wireless card has an additional 64 MB of memory for storing drivers and program files. We used 7 Stargate boards to implement the proposed peer to peer protocols (algorithms). The Stargate boards are configured to form a WLAN network, where in each node (Stargate board) acts like a peer. All the seven nodes form a peer to peer wireless Ad-hoc

network. We installed the symmetric java program into each peer using the HyperTerminal utility connected to the host machine. The Stargate system WLAN is assumed to be a single collision domain. Auctions are generated by separate nodes and transmitted to all the other nodes.

This chapter is organized as follows, Section 4.2 discusses the Stargate embedded system, Section 4.3 discusses the configuration of Stargate devices to form an Ad-hoc network i.e. Stargate WLAN configuration. Section 4.4 overviews the software tools used in this thesis. Section 4.5 gives a brief description of the Experimental methodology and the parameters used.

4.2 Stargate Embedded System

Stargate is a high performance processing platform designed for sensors, signal processing, control, and wireless sensor networking applications. The Stargate is preloaded with Linux and basic device drivers. Figure 4.1 shows a Stargate board, it has the following components:

- 32-bit 400 MHz, Intel PXA 255 Intel Xscale RISC Processor.
- 32 MB Flash and 64 MB SDRAM.
- RS 232 serial port that by default displays screen output to the terminal program at the other side.
- A 10/100 Ethernet that can provide wired LAN connectivity.
- A USB host that provides connections to USB devices when their drivers are installed.
- Wireless LAN Ambicom 802.11b CF card that provides wireless connectivity.

- JTAG that provides additional connectivity and data transfer features.

The Stargate system also has a daughter board that contains sockets for the wireless card and the Ethernet interface. The Stargate processor board has a wide variety of applications such as:

- A single-board computer running embedded Linux OS.
- A sensor network gateway.
- A customizable 802.11a/b wireless gateway.
- A cellular wireless gateway.
- Robotics controller card.
- Distributed computing platform.
- Embedded sensor signal processing unit.

Figure 4.1 : The Stargate board [27]

Before we begin the experiment, we need to connect our target Stargate boards to our host machine (a Linux or Windows PC). We used a windows host machine. To connect the target board to the host, we attach a null modem serial cable between the target Stargate board and an available serial port on the development environment (windows host). HyperTerminal is a terminal emulation and modem interface program included with windows. It is used to communicate with the Stargate board. COM Port parameters are selected in the HyperTerminal popup window, as shown in Figure 4.2

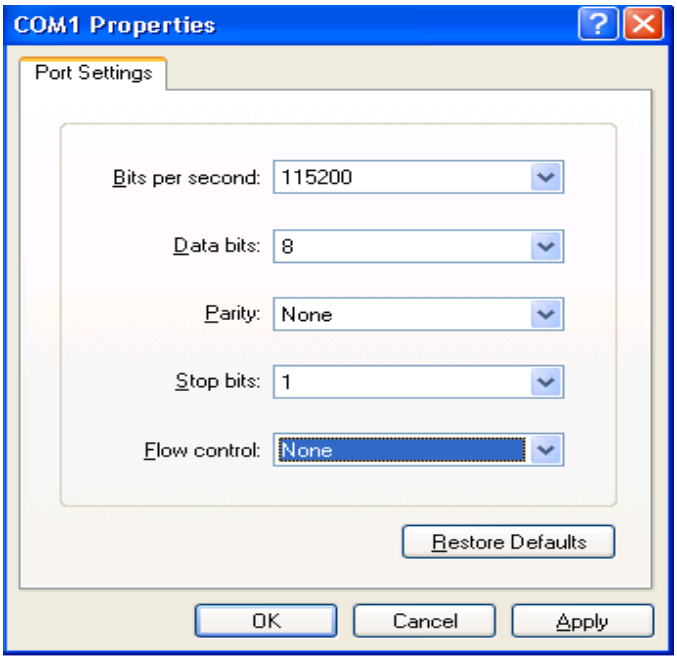

Figure 4.2: COM Port Settings

4.3 Stargate WLAN configuration

We can configure Stargate devices to form a WLAN manually, or we can automate the task, i.e., whenever we start the devices we need to assign the ip address. If we are configuring the devices manually, and if we are using automation in assigning the ip addresses to the devices, then we don't need to assign the ip address each time the

devices starts. To configure Stargate boards to form a WLAN network manually, following command is issued on each stargate board:

root# ifconfig wlan0 11.0.0.x

where *x* is any number between 1 to 7.

To provide a short-to-medium range, high speed remote access link to the Stargate, the advanced Stargate kit ships with the Ambicom wireless 802.11 card. This card plugs into the available PCMCIA slot using the PCMCIA adaptor module. Although this is a compact flash card, the CF slot is already allocated by the CF memory card through the adaptor.

The Stargate is preconfigured to recognize this card and to automatically load the required device drivers. However, some additional configuration is required to allow this card to join a network using the Ad-hoc or access point based methods. There is a file named wireless.opts located in the /etc/pcmcia folder that allows us to specify the settings for our wireless 802.11 network.

Each Stargate should be booted and configured manually for the very first time it is connected to the proposed wireless network, but subsequently we can automate the network setup using the method described in this section. The Stargate boards can be automated to form a WLAN by configuring two files: /etc/pcmcia/wireless.opts and /etc/opts/network.opts, present in the Stargate device. The wireless.opts includes configuration for an Ambicom wireless card. We can specify the name of the wireless network in the essid field and the type of the wireless network in the mode field. At present we set essid = robotics and mode = Ad-Hoc to indicate an Ad-hoc wireless

network having the name *robotics*, the remaining items are fixed for Ambicom wireless cards. The same wireless.opts file should be present on all Stargate boards present in the network. The network.opts file sets the network parameters that are specific to each Stargate board and need to be configured for each Stargate board separately on the host PC. The network.opts file sets DHCP = 0, because there is no DHCP in our network. It also sets default gateway to 11.0.0.1 i.e. the Stargate board acts as the router. The field IPAddr = 11.0.0.x where x should be distinct for each board. Both files can be edited on the host PC and sent to each Stargate board through the following commands:

scp wireless.opts root@11.0.0.x:/etc/pcmcia and

scp network.opts root@11.0.0.x:/etc/pcmcia

4.4 Overview of Software Tools

We use JAVA as our coding platform to test the functionality of our proposed algorithms, since Java supports multithreading. Within our Java program, multiple threads exist, with the following properties:

- Each thread executes code from its starting location in an ordered, predefined sequence, for a given set of inputs, threads have a common purpose, always executing the next statement in the sequence.
- Each thread executes its code independently of the other threads in the program.
- The threads have access to various types of data. Each thread is separate, so that local variables in the methods that the thread is executing are separate for different threads. These local variables are completely private; there is no way for one thread to access the local variables of another thread. If two threads happen to execute the same

method, each thread gets a separate copy of the local variables of that method. This is completely analogous to running two copies of the text editor, where each process would have separate copies of the local variables.

- Objects and their instance variables, on the other hand, can be shared between threads in a Java program, and sharing these objects between threads of a Java program is much easier than sharing data objects between processes in most operating systems.
- Static variables are automatically shared between all threads in a Java program.

The Java Run Time (JRE) is installed in the directory /mnt/cf1 that resides in the Flash card of the Stargate. A short-cut should be added to the /usr/sbin directory to avoid export path problems. The method for this is described in the Stargate developer's manual, and is also written below:

ln –s /mnt/cf1/jre /usr/sbin/jre

We used JCreator, which is a powerful interactive development environment (IDE) for Java technologies, to write our code for the proposed algorithms, and to compile. Writing and compiling of code is done on the windows host machine. After compilation, JCreator creates class files, which were transferred to each Stargate device through the windows HyperTerminal. The software on the Stargate comprises of a Java Runtime Environment (JRE). After transferring the class files we run these on each Stargate device by using the following command:

root# jre "filename".class

4.5 Experimental Methodology and Parameters used

All the experiments were carried out in the Corridor of the Computer Engineering Department of King Fahd University of Petroleum and Minerals. While conducting experiments it was kept in mind that no external interference occurs. Each experiment was conducted many times to ensure that the results are reproducible. The complete experimental procedure is explained below:

- First we transferred all the class files of the first algorithm (UDP P2P Reliable Broadcast with Token Passing) on each Stargate device through the HyperTerminal.
- Second symmetric java program is transferred to all the Stargate devices, as a result each node acts like a peer.
- Third we formed an Ad-hoc network by configuring each Stargate device. Here we used 7 Stargate devices to form an Ad-hoc network.
- Fourth we made sure that each Stargate is able to ping others.
- Fifth, we started the Main Module in each Stargate device.

Table 4.1 shows the Experimental parameters that we have set in order to evaluate the proposed protocols.

Parameter	Value	Meaning & Explanation
Timer	100 ms	Waiting time before broadcasting an Auction
Probability (p)	0.1	Broadcasting an Auction with the probability ‘P’ by each node
T(Ack)	7 ms	Minimum waiting time to send an Ack to the Sender
Port Number	2222	Port used for communication
Experimental Area	2m x 15m	This is the Area in which Experiments are conducted
Number of Nodes	7	Number of nodes used in the experiments
Speed	0.5m/s	Speed of the nodes when they are mobile

Table 4.1: Experimental Parameters used to evaluate the proposed protocols

Each Stargate node is coded to do 1000 Auctions in each experiment. Each experiment was conducted many times to ensure that the results are reproducible. We then observed the Auction time in each case and calculated the average Auction time for the number of times the experiment is performed. The results are shown in graphical form. We repeated the above steps for second algorithm (UDP P2P Reliable Distributed Broadcast).

CHAPTER 5

PERFORMANCE EVALUATION & COMPARITIVE STUDY

5.1 Introduction

This chapter presents the experimental results & performance evaluation of the proposed UDP P2P Reliable Broadcast with Token Passing and UDP P2P Reliable Distributed Broadcast schemes. These Protocols were coded in JAVA language. The performance gains of these protocols over previous work done in [3], are also documented in this chapter. The following sections discuss and elaborate on the results and performance of the two proposed protocols.

The organization of this chapter is as follows. Sections 5.2 and 5.3 provide analysis and discussion of results obtained for UDP P2P Reliable Broadcast with Token Passing and UDP P2P Reliable Distributed Broadcast schemes. Section 5.4 provides the Impact of mobility of devices on each proposed scheme. Section 5.5 gives the comparison of the proposed and the previous schemes found in the literature. Finally, Section 5.6 gives the comparison of the power consumption of the proposed and the previous schemes found in the literature.

5.2 UDP P2P Reliable Broadcast with Token Passing scheme

This section provides analysis and discussion of results obtained for UDP P2P Reliable Broadcast with Token Passing scheme.

Figure 5.1 below shows the distribution of completion times of Auctions. Auction time in milliseconds is taken on the horizontal axis and percentage of cases (percentage of cases the Auction times are occurring out of 1000 Auctions) is taken on the vertical axis.

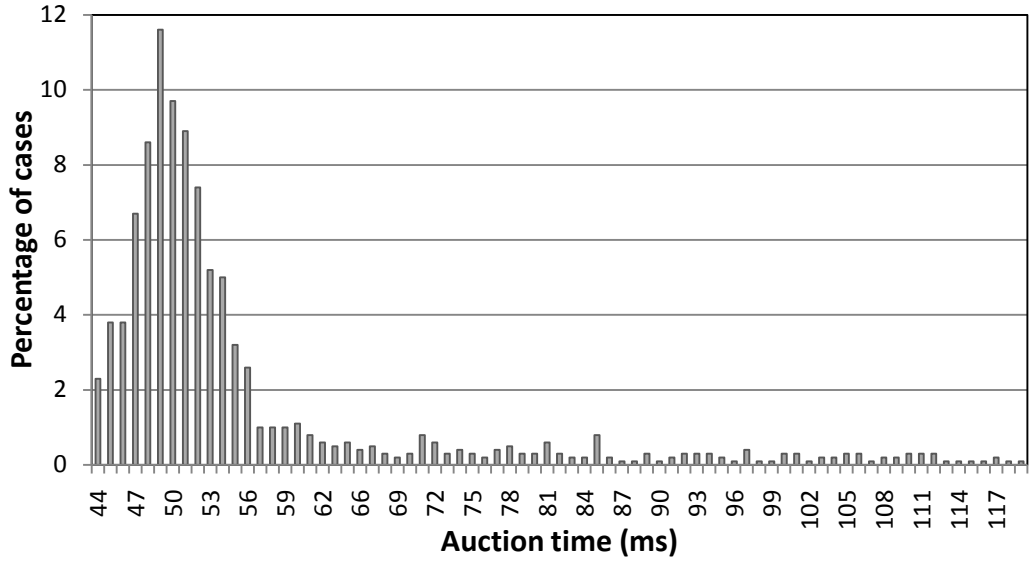

Figure 5.1: Distribution of Completion times of Node7

The histogram distribution of completion times of node 7 for peer to peer Auction using UDP P2P Reliable Broadcast with Token passing scheme is shown in the above plot (Figure 5.1) with N=1000 i.e, 1000 auctions by the node 7. For each auction time (t), the plot displays a column that corresponds to the percentage of cases for which the auctioning time was measured as t. It is clear that in 80% of the experimented cases, the average auction times fall below 57ms. The distribution also shows the scattering within

the range 44-117ms with most of the concentration within the range [44, 57] ms. The average auctioning time is 56ms. From the above plot, we can see that around 80% of the auctions are within the range of 44-57ms, and the rest of the 20% of the auctions are scattered beyond 57ms. The histogram distribution of completion times of remaining nodes for peer to peer Auction using UDP P2P Reliable Broadcast with Token Passing scheme are given in Appendix A.

Table 5.1 shows the summary of Average Auction Time, Standard Deviation and Range for true population mean of each node. As we can see, there is not much deviation in the Average Auction Time of each node.

Node	**Average Auction Time(ms)**	**Standard Deviation**	**Range for true population mean**
Node 1	61	16.9	[60.9,63.0]
Node 2	57	15.0	[56.1,57.9]
Node 3	57	15.2	[56.1,57.9]
Node 4	59	16.1	[58.8,60.8]
Node 5	56	15.4	[56.0,57.9]
Node 6	57	15.0	[56.1,57.9]
Node 7	56	14.7	[55.1,56.9]

Table 5.1: Summary of the Average Auction Time, Standard Deviation and Range for true population mean of each node (UDP P2P Reliable Broadcast with Token Passing Scheme)

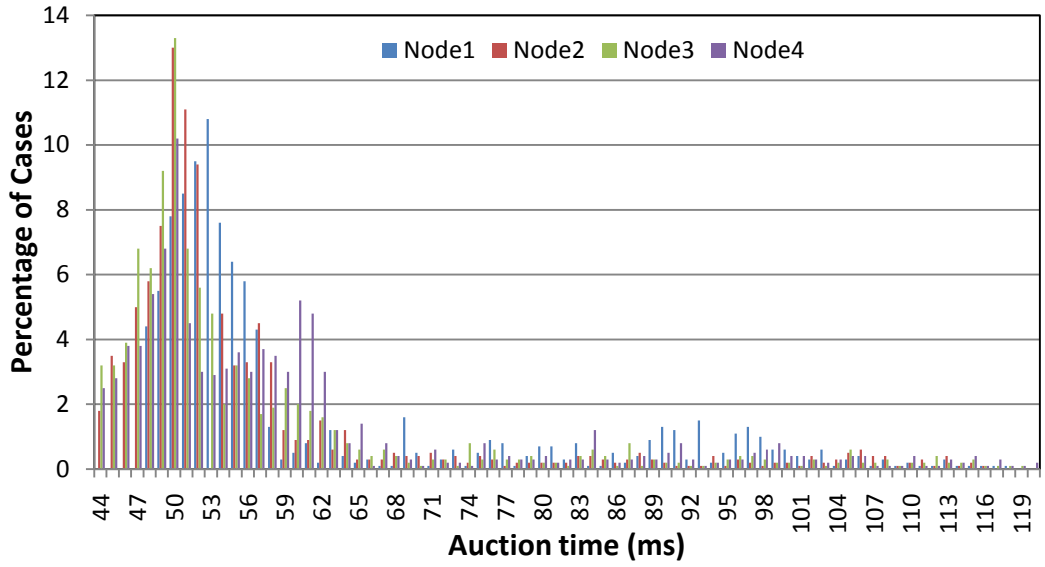

Figure 5.2: Distribution of completion times of first four nodes (1, 2, 3, and 4)

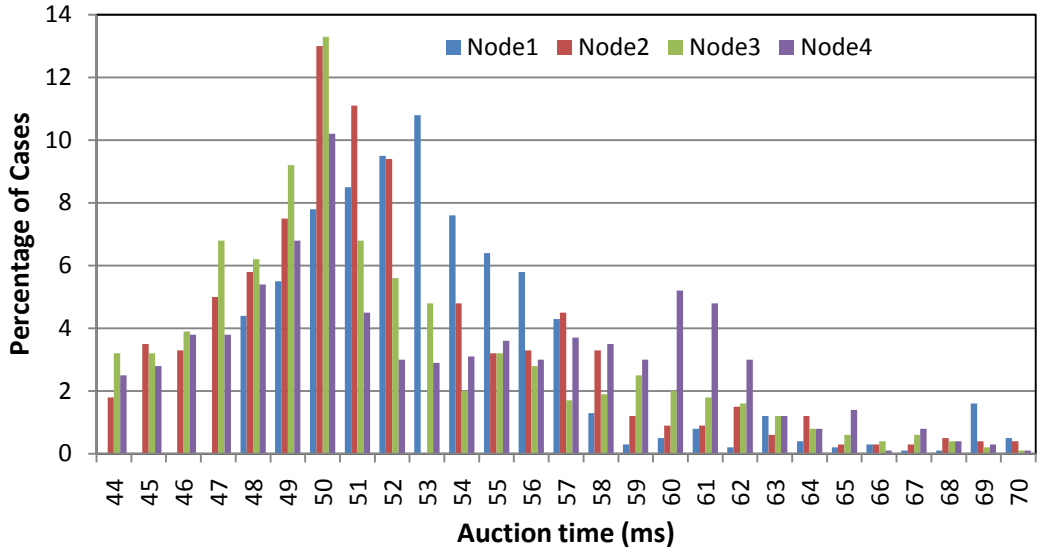

Figure 5.3: Distribution of completion times of first four nodes showing Auction time between 44-70ms

Figure 5.2 and Figure 5.3 represent the histogram distribution of overall auction completion times of the first four nodes. Each node had the opportunity to generate 1000

auctions, i.e. a total of 4000 auctions. The plot shows the percentage of auctions that falls into the reported time (ms). For each auction time (t) the plot displays a set of columns (1 to 4), each column corresponding to the percentage of cases for each node for which the auctioning time was measured as t. It is clear that in the large majority of experimented cases, the average auction times fall below 58 ms. The distribution also shows the scattering within the range 44-119ms with most of the concentration within the range 44-59ms. Figure 5.3 is same as Figure 5.2 except that it shows the plot for Auction time from 44 to 70ms, to get the clear picture of the Auctioning done by each node.

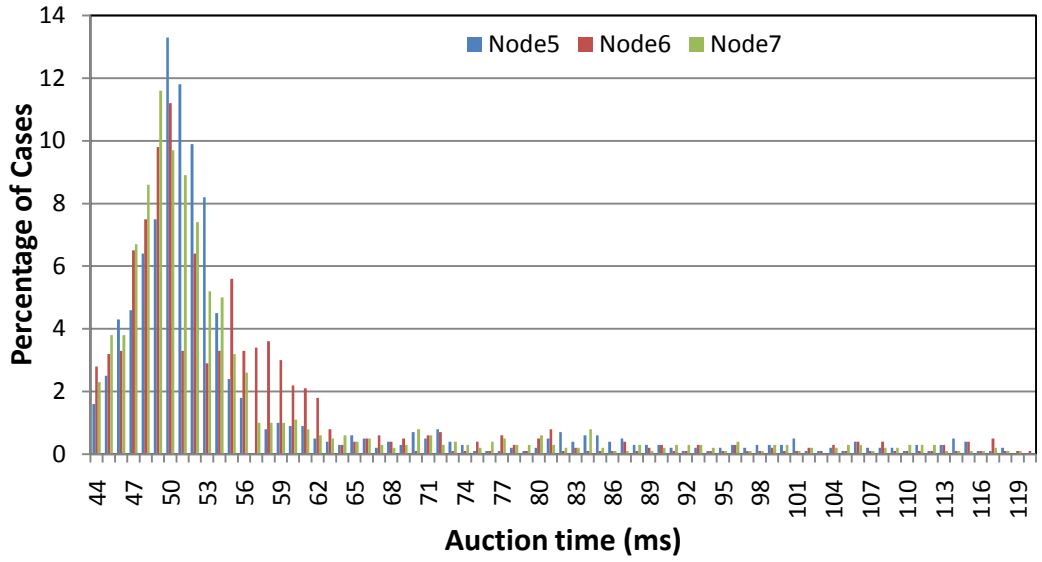

Figure 5.4: Distribution of completion times of last three nodes (5, 6 and 7)

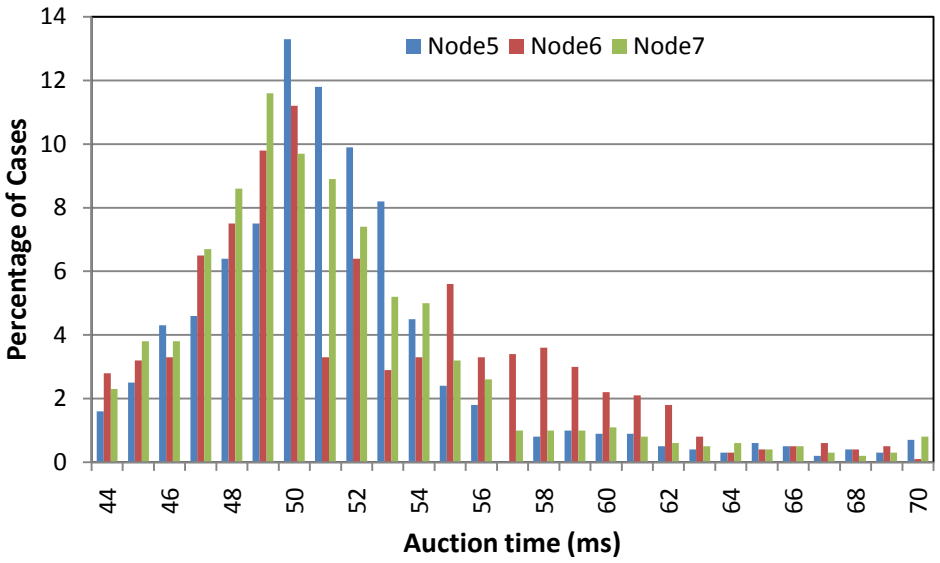

Figure 5.5: Distribution of completion times of last three nodes showing Auction time between 44-70ms

Figure 5.4 and Figure 5.5 represent the histogram distribution of overall auction completion times of the last three nodes. Each node had the opportunity to generate 1000 auctions, i.e. a total of 3000 auctions. The plot shows the percentage of auctions that falls into the reported time (ms). For each auction time (t), the plot displays a set of columns (1 to 3), each corresponding to the percentage of cases for each node for which the auctioning time was measured as t. It is clear that in the large majority of experimented cases the average auction times is below 58ms. The distribution also shows the scattering within the range 44-119ms with most of the concentration within the range 44-56ms. Figure 5.5 is same as Figure 5.4, except that it shows the plot for Auction time from 44 to 70ms, to get the clear picture of the Auctioning done by each node.

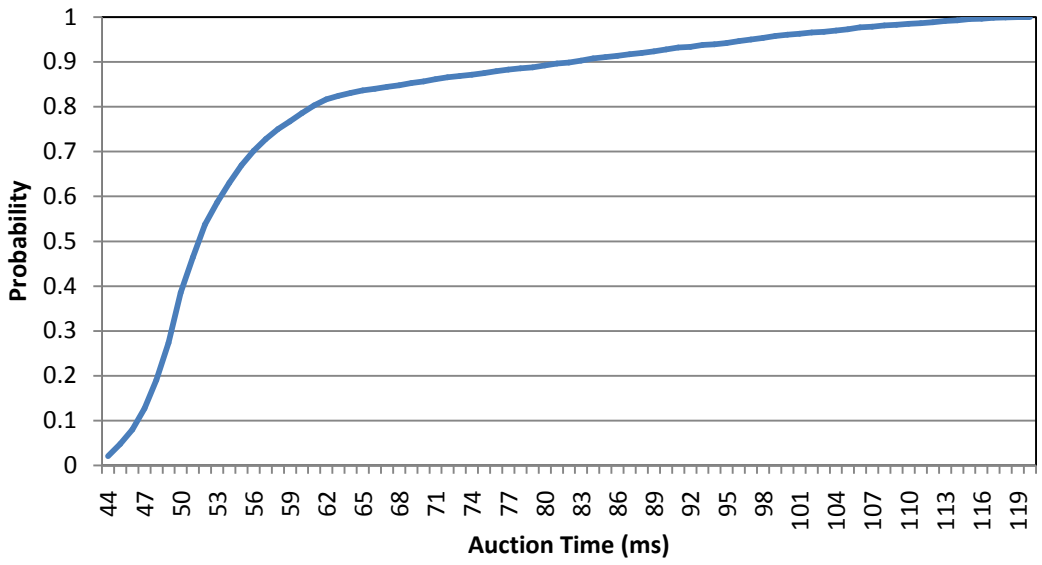

Figure 5.6: CDF of Auctions time of all the nodes averaged (UDP P2P Reliable Broadcast with Token Passing scheme)

The CDF of Auction times of all the nodes averaged using UDP P2P Reliable Broadcast with Token Passing scheme is shown in above Figure 5.6. It is clear that, 70% of the Auctions are completed in less than 55ms, 80% of the Auctions are completed in the range of 44-61ms, 90% of the Auctions are completed between 44 and 82ms, and 10% of the Auctions are completed in the range of 82-119ms.

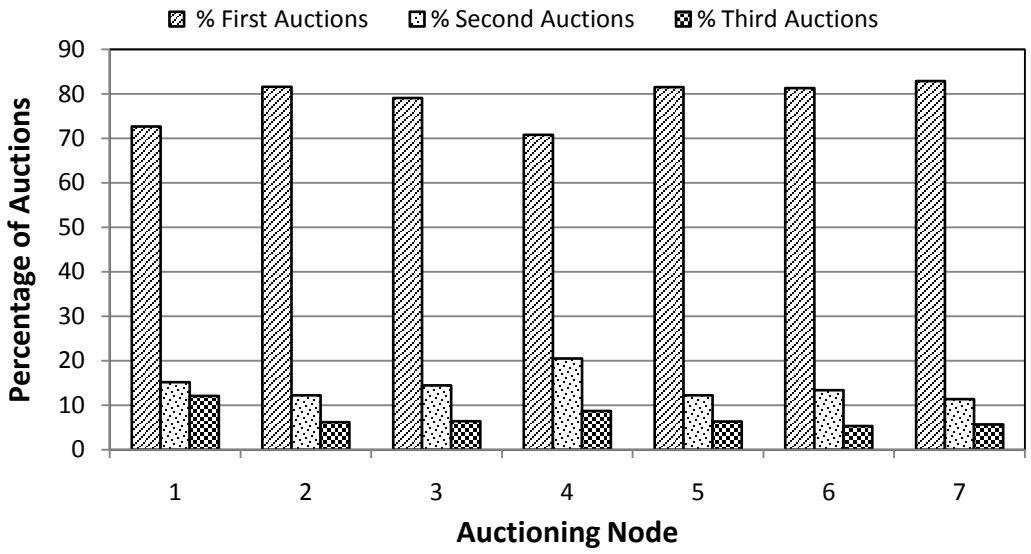

Figure 5.7: Reliability of P2P auctioning using UDP P2P Reliable Broadcast with Token passing scheme

The degree of reliability of the proposed protocol along with overall auctioning times shows that UDP P2P Reliable Broadcast with Token passing scheme appears to be reliable, as all experienced auctions among 7 stargate nodes have been completed using only three auctioning steps. Figure 5.7 shows the Percentage of Auctions by each node, i.e. Percentage of Auctions completed in the First attempt, the Second attempt, or the Third attempt. In 80%, 12%, and 8% of the cases all the nodes responded after the first auction, the second auction and the third auction respectively. One needs three auctions to make sure all the nodes of our Ad-hoc network have been reached and successfully replied during the auction.

5.3 UDP P2P Reliable Distributed Broadcast scheme

This section provides analysis and discussion of results obtained for the UDP P2P Reliable Distributed Broadcast scheme.

Figure 5.8 below shows the distribution of the completion times of Auctions. Auction time in milliseconds is taken on the horizontal axis and percentage of cases (percentage of cases the Auction times are occurring out of 1000 Auctions) is taken on the vertical axis.

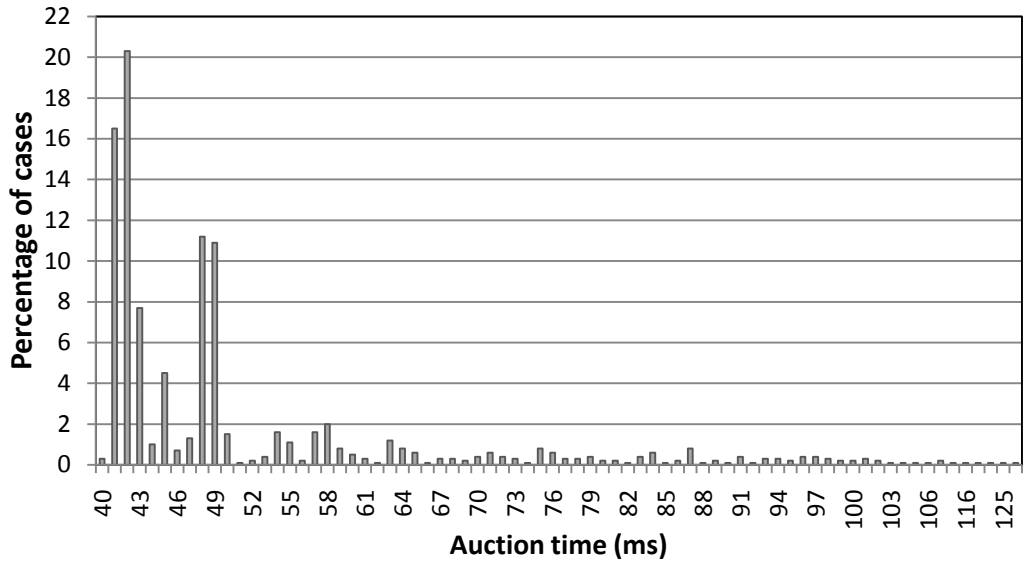

Figure 5.8: Distribution of Completion times of Node3

The Histogram distribution of completion times of node 3 for peer to peer Auction using UDP P2P Reliable Distributed Broadcast scheme is shown in the above plot (Figure 5.8) with N=1000 i.e., 1000 auctions by node 3. For each auction time (t), the plot displays a column that corresponds to the percentage of cases for which the auctioning time was measured as t. It is clear that in 76% of the experimented cases the average auction times remain below 50ms. The distribution also shows the scattering within the range 40-126ms with most of the concentration within the range 40-50ms. The average auctioning time is 52ms. From the above plot, we can see that around 76% of the auctions are within the range of 40-50ms, and the rest of the 24% of the auctions are scattered beyond 50ms.

The histogram distribution of completion times of remaining nodes for peer to peer Auction using UDP P2P Reliable Distributed Broadcast scheme are given in Appendix A.

Table 5.2 shows the summary of Average Auction Time, Standard Deviation and Range for true population mean of each node. As we can see, there is not much deviation in the Average Auction Time of each node.

Node	**Average Auction Time(ms)**	**Standard Deviation**	**Range for true population mean**
Node 1	55.4	18.8	[54.2,56.6]
Node 2	52.2	18.2	[51.1,53.3]
Node 3	52.1	15.8	[51.1,53.1]
Node 4	55.4	19.1	[54.2,56.6]
Node 5	52.0	18.5	[50.8,53.1]
Node 6	52.2	16.4	[51.2,53.2]
Node 7	52.6	16.5	[51.6,53.6]

Table 5.2: Summary of the Average Auction Time, Standard Deviation and Range for true population mean of each node (UDP P2P Reliable Distributed Broadcast)

5.3.1 Distribution of Completion times of Auctions of all 7 nodes

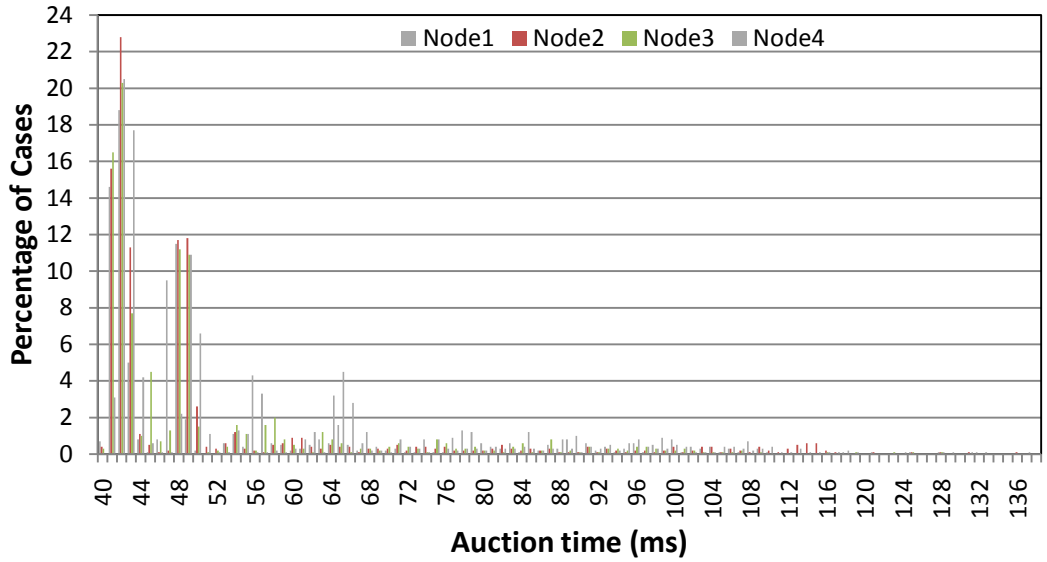

Figure 5.9: Distribution of completion times of the first four nodes (1, 2, 3, and 4)

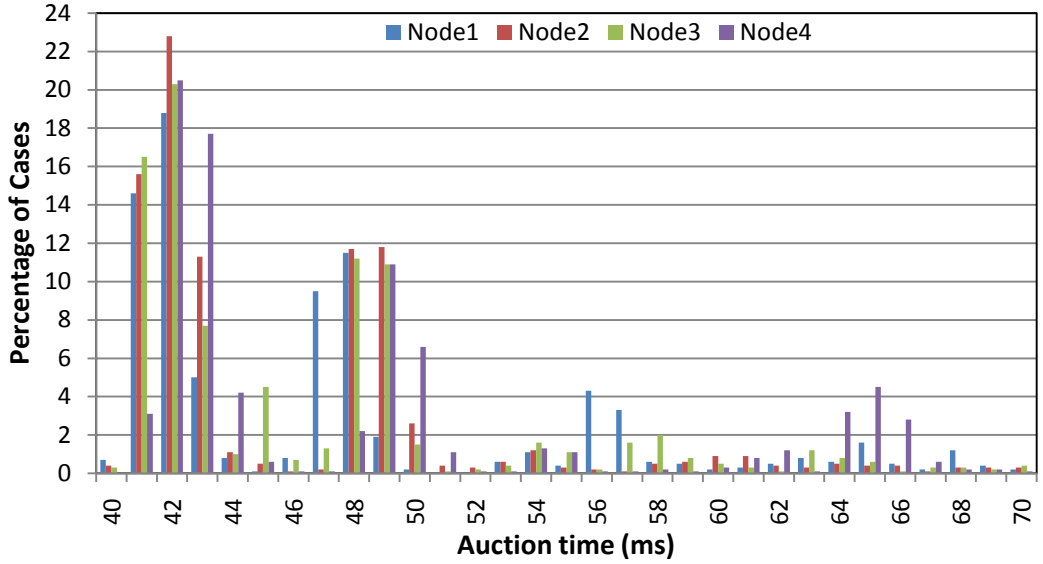

Figure 5.10: Distribution of completion times of the first four nodes showing Auction time between 40-70ms

Figure 5.9 and Figure 5.10 represent the histogram distribution of auction overall completion times of the first four nodes. Each node had the opportunity to generate 1000 auctions, i.e. a total of 4000 auctions. The plot shows the percentage of auctions that fall into this reported time (ms). For each auction time (t), the plot displays a set of columns (1 to 4), each corresponding to the percentage of cases for each node for which the auctioning time was measured as t. It is clear that in the large majority of experimented cases the average auction times remain below 50ms. The distribution also shows the scattering within the range 40-139ms with most of the concentration within the range 40-50ms. Figure 5.10 is same as the Figure 5.9 except that it shows the plot for Auction time between 40 and 70ms, to get a clear picture of the Auctioning done by each node.

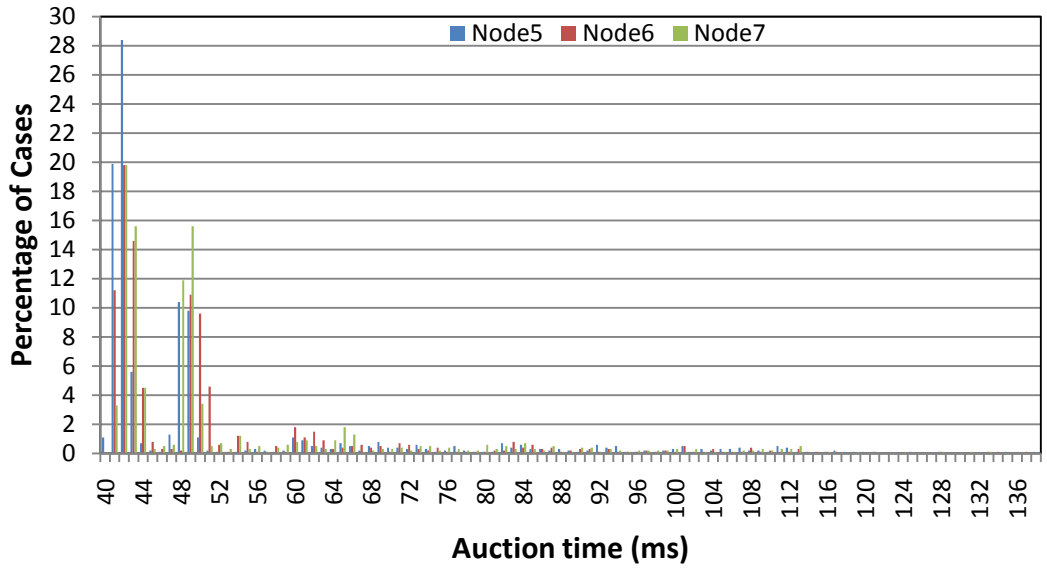

Figure 5.11: Distribution of completion times of the last three nodes (5, 6 and 7)

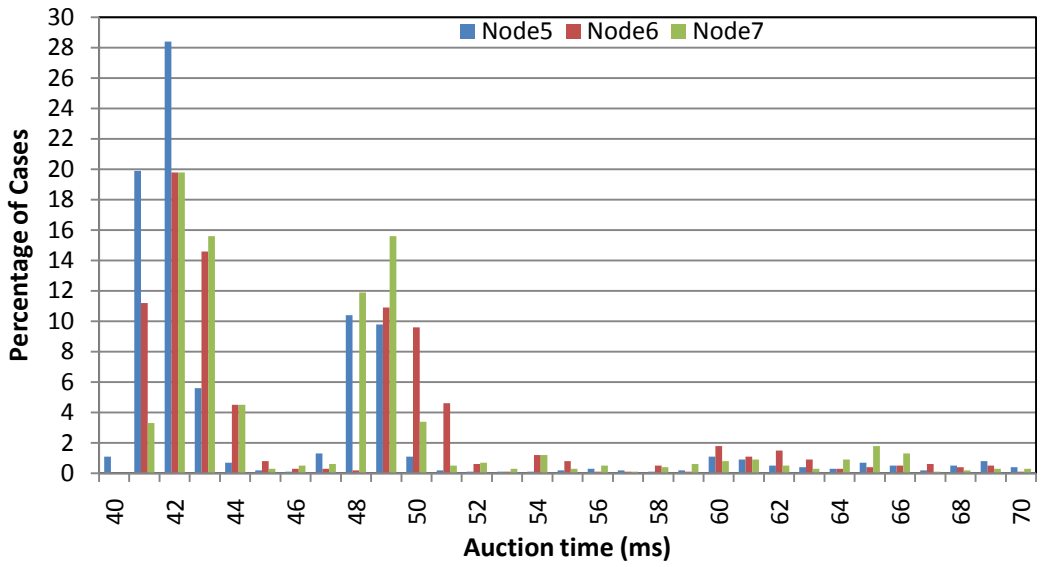

Figure 5.12: Distribution of completion times of the last three nodes showing Auction time between 40-70ms

Figure 5.11 and Figure 5.12 represent the histogram distribution of auction overall completion times of the last three nodes. Each node had the opportunity to generate 1000 auctions, i.e. a total of 3000 auctions. The plot shows the percentage of auctions that fall into this reported time (ms). For each auction time (t), the plot displays a set of columns (1 to 3), each corresponds to the percentage of cases for each node for which the auctioning time was measured as t. It is clear that in the large majority of experimented cases the average auction times remain below 51ms. The distribution also shows the scattering within the range 40-136ms with most of the concentration within the range 40-51ms. Figure 5.12 is same as the Figure 5.11, except that it shows the plot for Auction time between 40 and 70ms, to get a clear picture of the Auctioning done by each node.

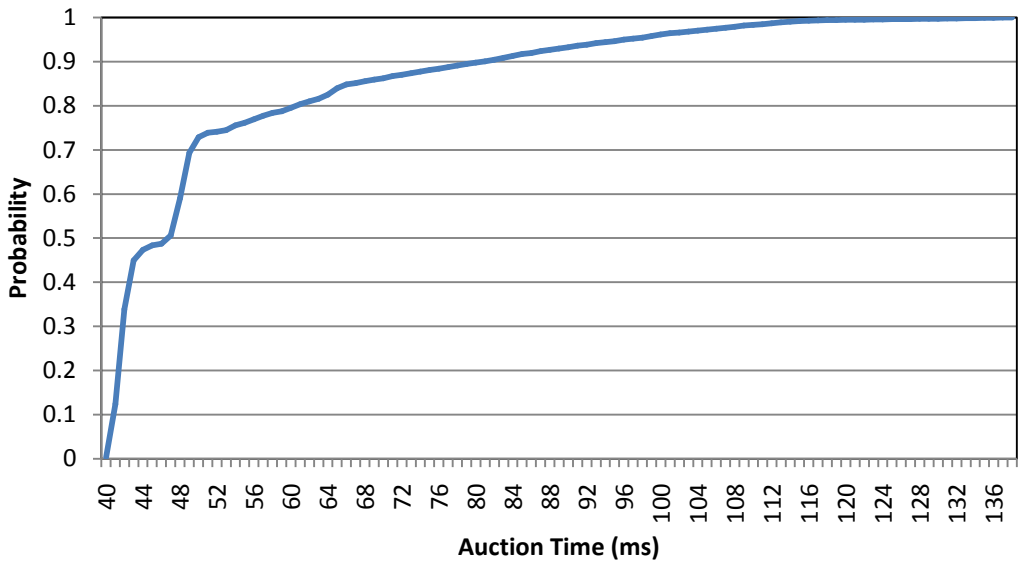

Figure 5.13: CDF of Auctions times of all the Nodes averaged (UDP P2P Reliable Distributed Broadcast scheme)

The CDF of Auction times of all the nodes averaged using UDP P2P Reliable Broadcast with Token Passing scheme is shown in above Figure 5.13. It is clear that, 70% of the Auctions are completed in less than 49ms, 80% of the Auctions are completed in the range of 40-60ms, 90% of the Auctions are completed between 40 and 80ms, and 10% of the Auctions are completed in the range of 80-138ms.

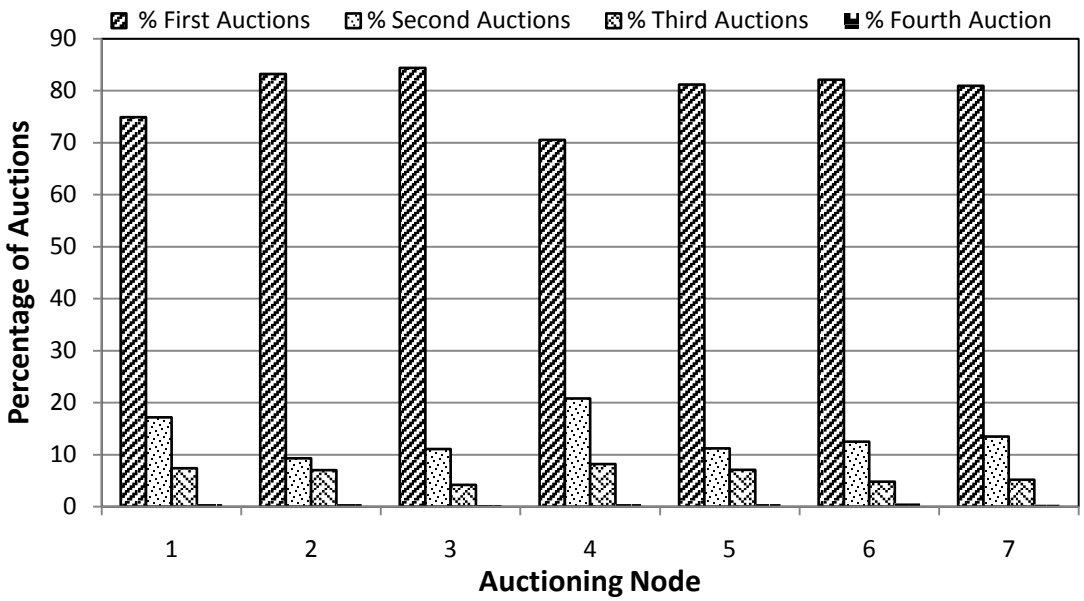

Figure 5.14: Reliability of peer to peer auctioning using UDP P2P Reliable Distributed Broadcast scheme

The degree of reliability of the proposed protocol altogether with overall auctioning times show that UDP P2P Reliable Distributed Broadcast scheme appears to be reliable, as all experienced auctions among 7 stargate nodes are completed using only four auctioning steps. Figure 5.14 shows the Percentage of Auctions by each node, i.e. Percentage of Auctions completed in the First attempt, the Second attempt, the Third attempt, or the Fourth attempt. In 82%, 12%, 5.5% and 0.5% of the cases all the nodes responded after the first auction, the second auction, the third auction and the fourth auction, respectively. One needs four auctions to make sure that all nodes of our Ad-hoc network have been reached and have successfully replied back.

5.4 Impact of Mobility

To measure the impact of mobility on the proposed protocols, two mobility scenarios are considered. In scenario I, two nodes are mobile i.e. nodes 2 and 7. In scenario II, three nodes are mobile i.e. nodes 2, 3, and 7.

5.4.1 Impact of mobility on UDP P2P Reliable Broadcast with Token Passing scheme

Scenario I: When two nodes are mobile, i.e. nodes 2 and 7 are moving.

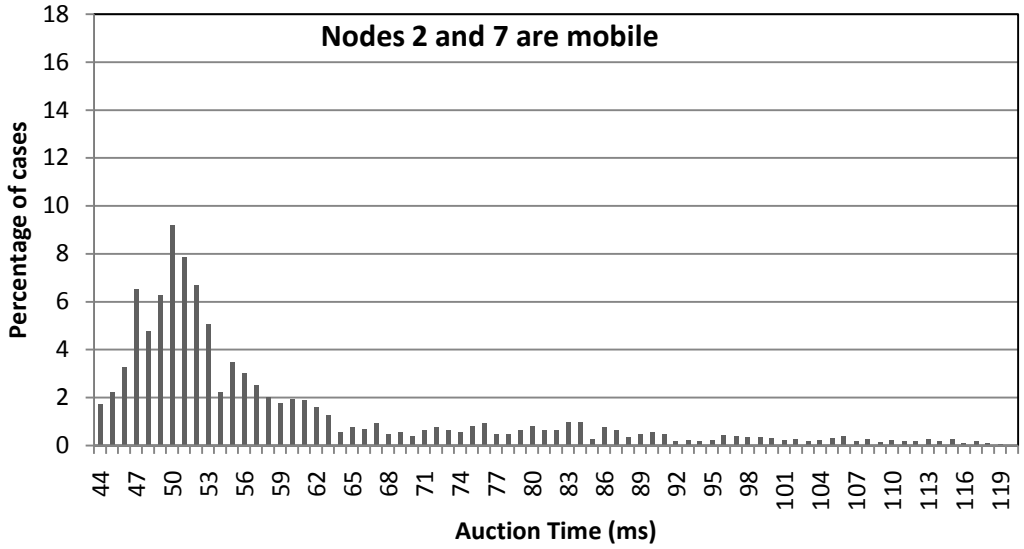

Figure 5.15: Distribution of Auction completion times (Averaging all Auction times for 7 nodes) when two nodes are mobile using UDP P2P Reliable Broadcast with Token Passing scheme

The histogram distribution of Auction completion times using UDP P2P Reliable Broadcast with Token Passing scheme when two nodes are mobile is shown in Figure 5.15. The distribution is of all the 7 nodes averaged, with N=7000 i.e., 7000 auctions by all the nodes. It is clear that in 82% of the experimented cases, the average auction times

fall below 62ms. The distribution also shows the scattering within the range 44-119ms. The average auctioning time is 60ms with standard deviation of 16.

Scenario 2: When three nodes are mobile, i.e. nodes 2, 3 and 7 are moving.

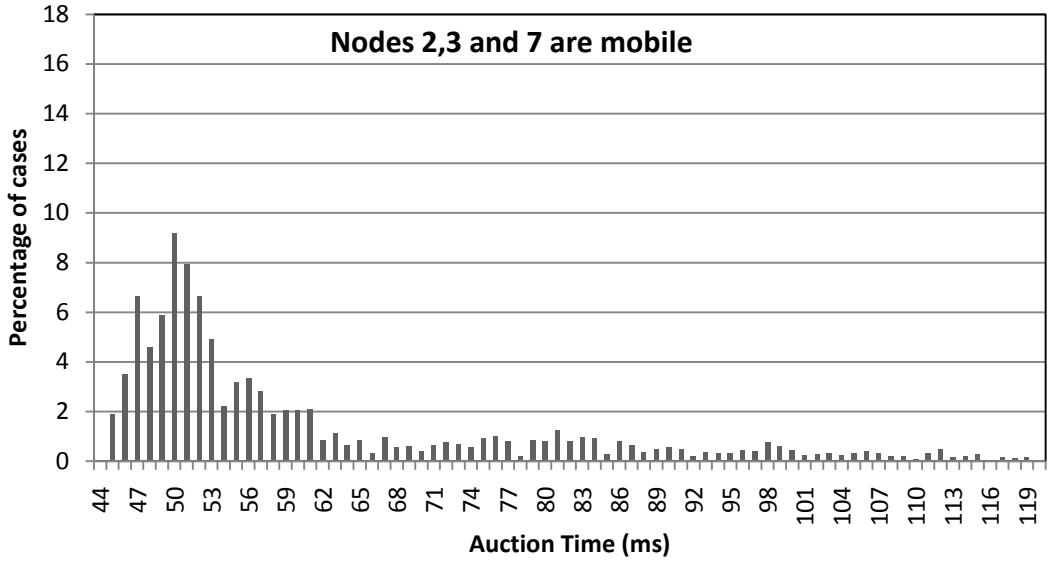

Figure 5.16: Distribution of Auction completion times (Averaging all Auction times for 7 nodes) when three nodes are mobile using UDP P2P Reliable Broadcast with token passing scheme

The histogram distribution of Auction completion times using UDP P2P Reliable Broadcast with Token passing scheme when three nodes are mobile is shown in Figure 5.16. The distribution is of all the 7 nodes averaged, with N=7000 i.e., 7000 auctions by all the nodes. It is clear that in 82% of the experimented cases, the average auction times fall below 61ms. The distribution also shows the scattering within the range 44-119ms. The average auctioning time is 60.4ms with a standard deviation of 17.

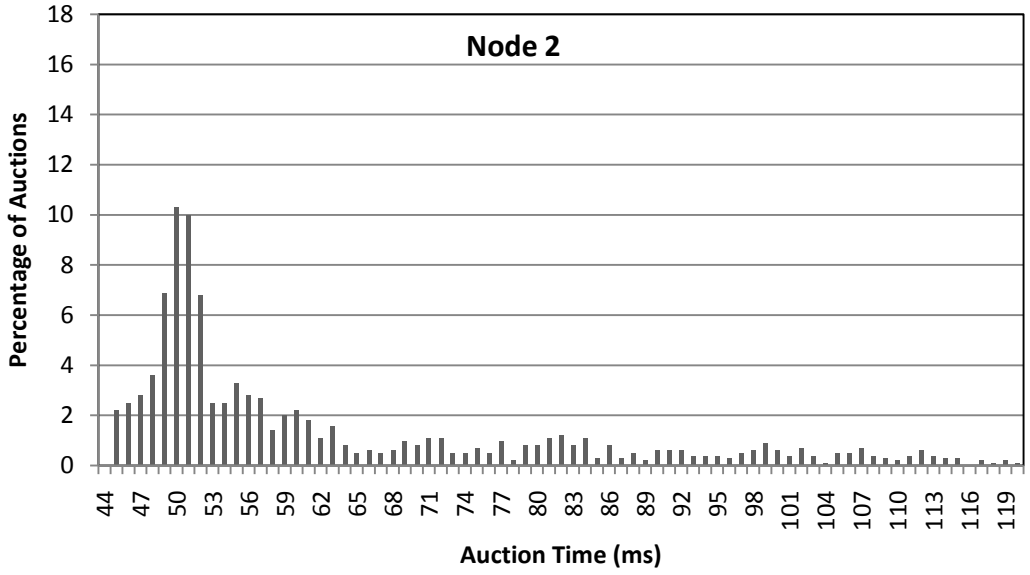

Figure 5.17: Distribution of Auction completion times of Node 2, when three nodes are mobile using UDP P2P Reliable Broadcast with token passing scheme

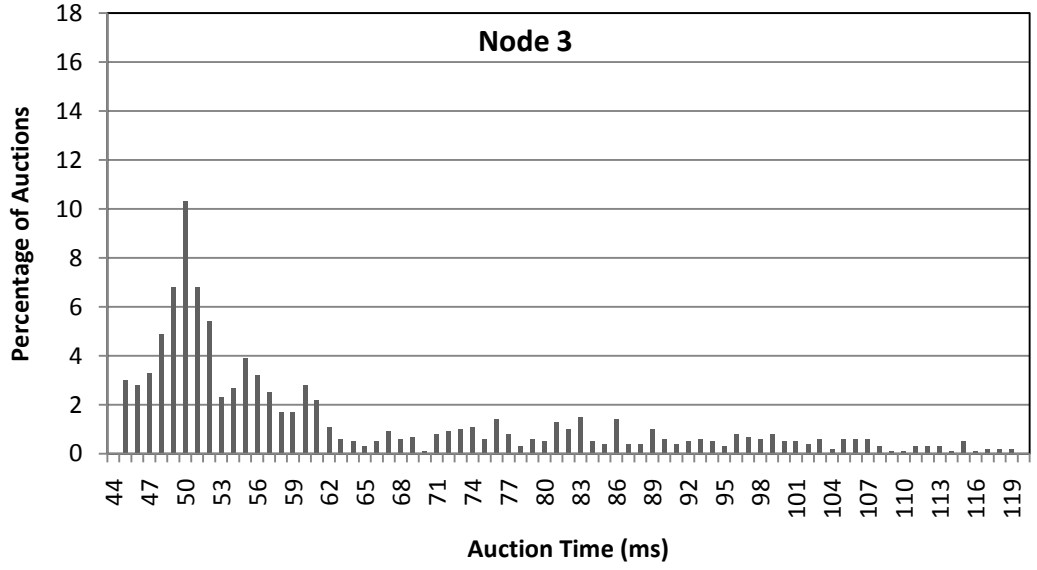

Figure 5.18: Distribution of Auction completion times of Node 3, when three nodes are mobile using UDP P2P Reliable Broadcast with token passing scheme

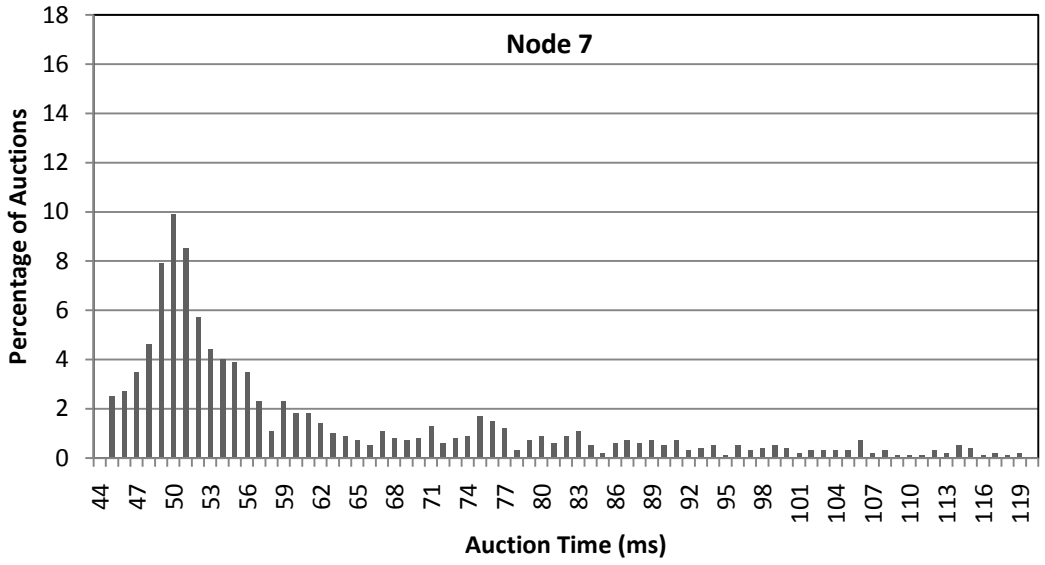

Figure 5.19: Distribution of Auction completion times of Node 7, when three nodes are mobile using UDP P2P Reliable Broadcast with token passing scheme

Figures Figure 5.17,Figure 5.18, and Figure 5.19 show the distribution of Auction completion times of nodes 2, 3, and 7 respectively i.e., these are the three nodes that are mobile. Euclidean distance is used to measure the similarity between the histograms. We used the following formula to calculate the Euclidean distance between nodes (2, 3), nodes (3, 7), and nodes (2, 7):

$$D = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_1 - q_1)^2} \quad (5.1)$$

where ‘p’ and ‘q’ are Euclidean vectors

Tables (Table 5.3 & Table 5.4) below show the Average Auction Time and Standard Deviation of each node i.e., node 2, 3, and 7, and Euclidean distance between nodes (2, 3), nodes (3, 7), and nodes (2, 7). While comparing Average Auction Time and Standard Deviation of each node, and Euclidean distance between nodes (2, 3), nodes (3, 7), and

nodes (2, 7) we can say that the performance patterns (histograms) of these nodes are symmetric.

Node	Average Auction Time(ms)	Standard Deviation
Node 2	62.9	18.3
Node 3	63.2	18.4
Node 7	61.5	16.9

Table 5.3: Average auction time and standard deviation of nodes 2, 3 and 7

d1(2&3)	d2(2&7)	d3(3&7)
47.6	44.4	43.3

Table 5.4: Euclidean distance between nodes (2, 3), nodes (3, 7), and nodes (2, 7)

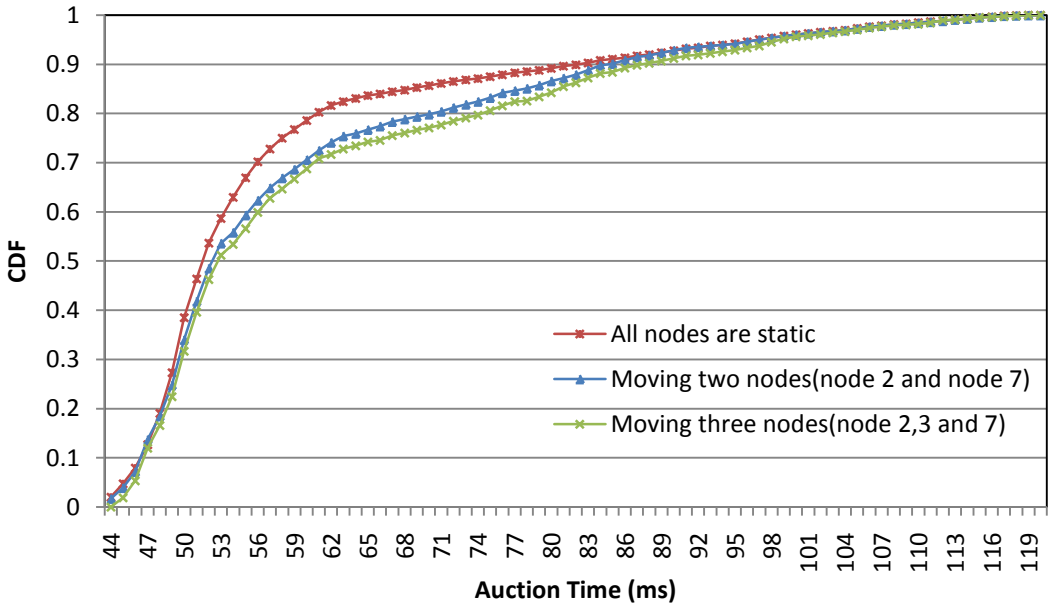

Figure 5.20: CDF's of Auction times of three different scenarios using UDP P2P Reliable Broadcast with token passing scheme

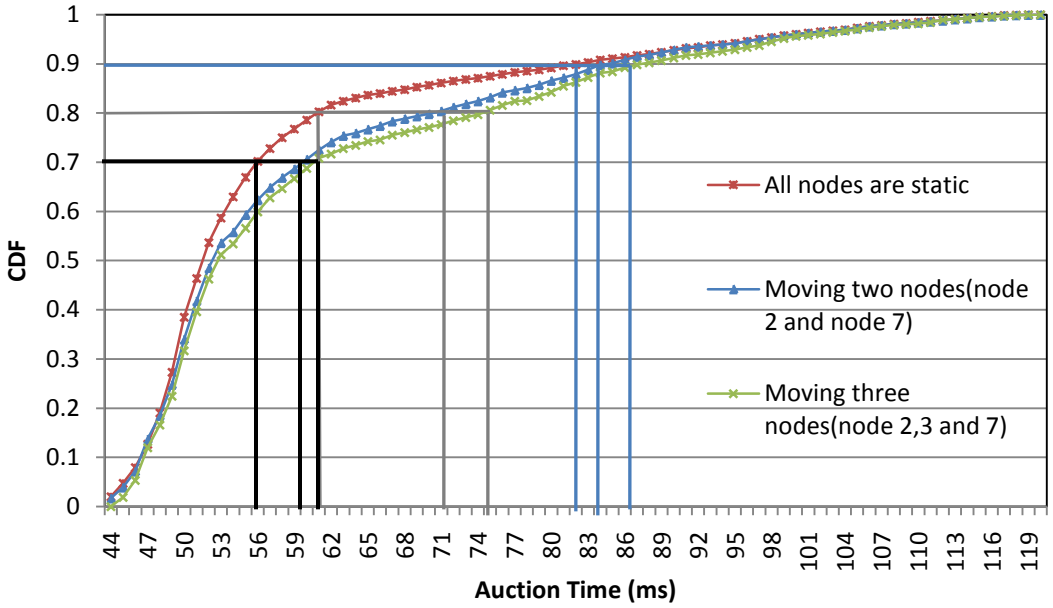

Figure 5.21: CDF's of Auction times of three different scenarios using UDP P2P Reliable Broadcast with token passing scheme

The CDF's of Auction times of three different scenarios i.e., CDF of Auction times when all the nodes are static, CDF of Auction times when two nodes are mobile, and CDF of Auction times when three nodes are mobile using UDP P2P Reliable Broadcast with Token Passing scheme is shown in Figure 5.20. When the nodes are mobile we notice an increase in disparity, as the number of nodes moving increases, the disparity also increases. It is clear from the above plot that, 70% of the Auctions are completed in less than 56ms when all the nodes are static, whereas it slightly increases to 59 ms, when two nodes are moving. When three nodes are moving, 70% of auctions are completed within 61ms. Moreover, 80% of the Auctions are completed in less than 61ms when all the nodes are static, whereas it slightly increases to 71 ms, when two nodes are moving. When three nodes are moving, 80% of the Auctions are within range 44-74ms, and 10%

of the Auctions are within the range 82-117ms when all the nodes are static, whereas it slightly decreases to 83-119ms when two nodes are moving. When three nodes are moving, 10% of Auctions are within the range 86-119ms.

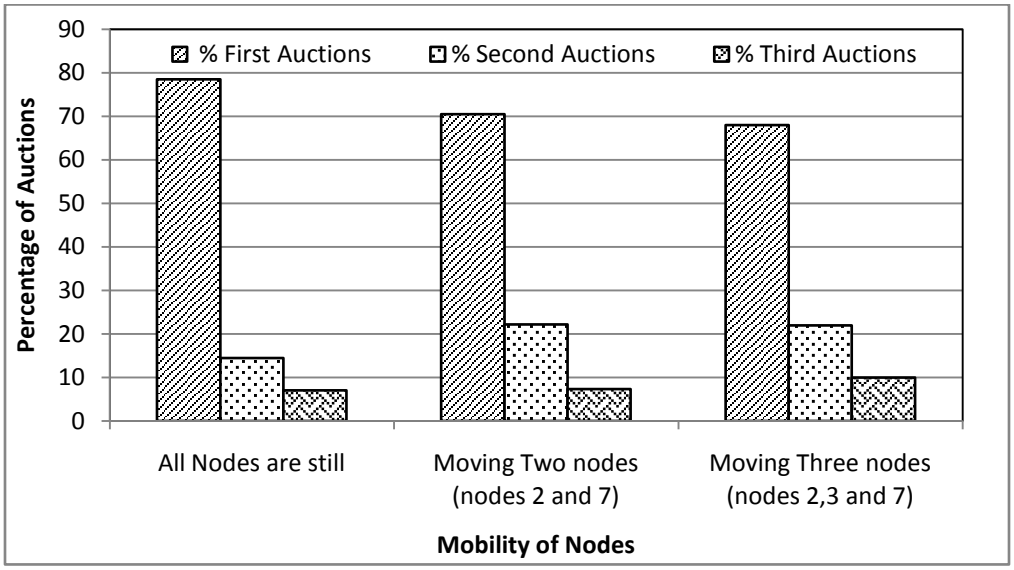

Figure 5.22: Reliability of UDP P2P Reliable Broadcast with token passing scheme with three different scenarios

Scenario	Average Auction Time (ms)	Standard Deviation (S.D)
All Nodes are static	58	15.6
Two Nodes are mobile	60	16
Three Nodes are mobile	60.4	17

Table 5.5: Summary of Average Auction Time (ms) and Standard Deviation of each scenario using UDP P2P Reliable Broadcast with Token Passing scheme

The UDP P2P Reliable Broadcast with Token Passing scheme is reliable, as all the experienced auctions among seven Stargate nodes are completed using only three auctioning steps. Figure 5.22 shows the Percentage of Auctions by all the nodes in three

different scenarios, i.e. Percentage of Auctions completed in the First attempt, the Second attempt, or the Third attempt. One needs three auctions to make sure that all the nodes of our Ad-hoc network have successfully replied back. Although, when 30% of the nodes are mobile (two nodes are moving), the Average auction time is increased by only 60ms with S.D = 16, and when 40% of the nodes are mobile (three nodes are moving), the Average auction time is increased by only 60.4ms with S.D = 17.

5.4.2 Impact of mobility on UDP P2P Reliable Distributed Broadcast scheme

Scenario 1: When two nodes are mobile i.e., nodes 2 and 7 are moving.

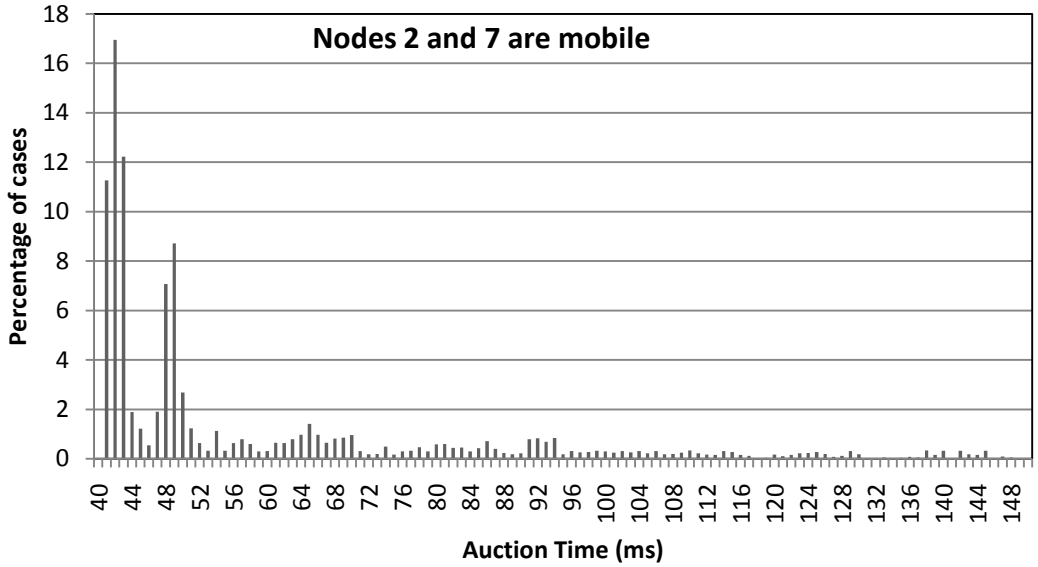

Figure 5.23: Distribution of Auction completion times (Averaging all Auction times for 7 nodes) when two nodes are mobile using UDP P2P Reliable Distributed Broadcast scheme

The histogram distribution of Auction completion times using UDP P2P Reliable Distributed Broadcast scheme when two nodes are mobile is shown in Figure 5.23. The distribution is of all the 7 nodes averaged, with N=7000 i.e., 7000 auctions by all the

nodes. It is clear that in 75% of the experimented cases the average auction times fall below 52ms. The distribution also shows the scattering is within the range 40-149ms. The average auctioning time is 58.3ms with standard deviation of 24.4.

Scenario 2: When three nodes are mobile i.e., nodes 2, 3 and 7 are moving.

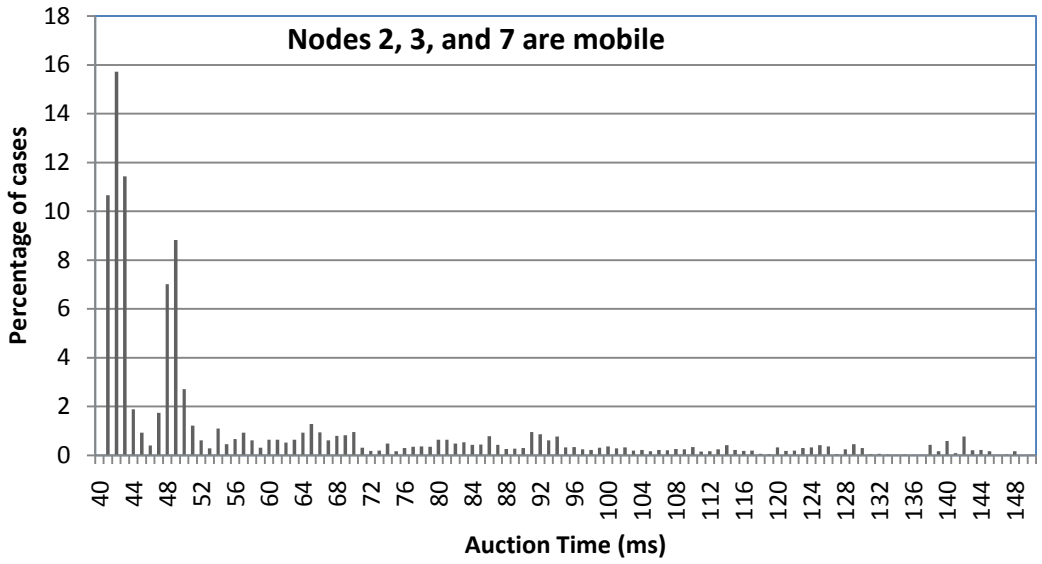

Figure 5.24: Distribution of Auction completion times (Averaging all Auction times for 7 nodes) when three nodes are mobile using UDP P2P Reliable Distributed Broadcast scheme

The histogram distribution of Auction completion times using UDP P2P Reliable Distributed Broadcast scheme when three nodes are mobile is shown in Figure 5.24. The distribution is of all the 7 nodes averaged, with N=7000 i.e., 7000 auctions by all the nodes. It is clear that in 75% of the experimented cases the average auction times fall below 52ms. The distribution also shows the scattering is within the range 40-149ms. The average auctioning time is 60.7ms with standard deviation of 26.6.

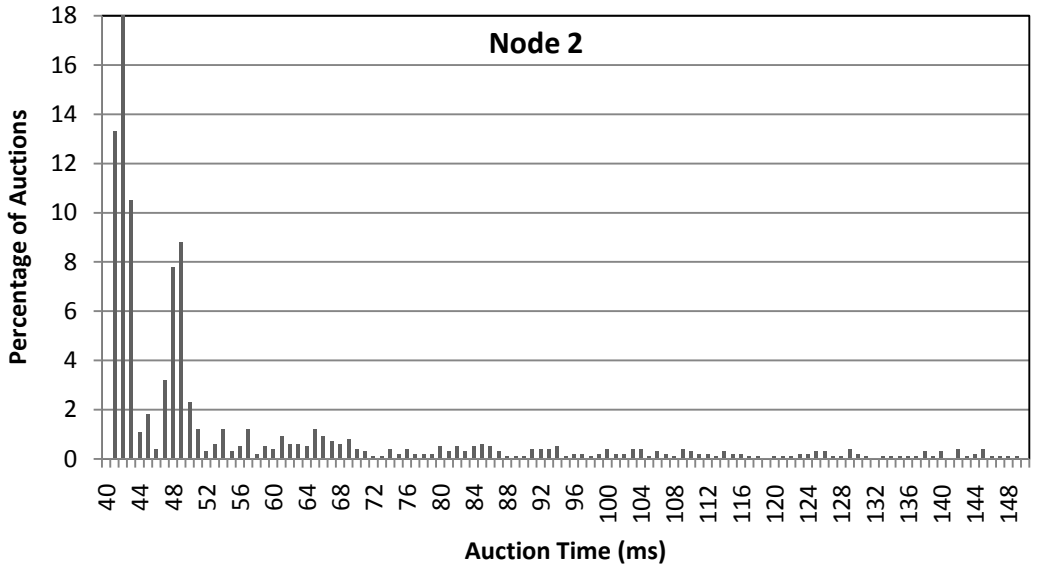

Figure 5.25: Distribution of Auction completion times of Node 2, when three nodes are mobile using UDP P2P Reliable Distributed Broadcast scheme

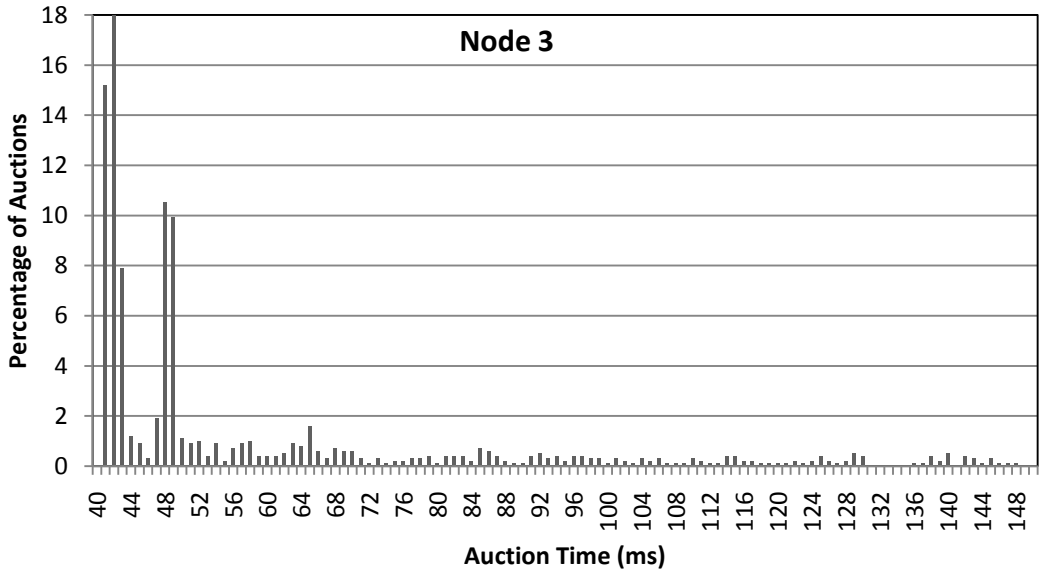

Figure 5.26: Distribution of Auction completion times of Node 3, when three nodes are mobile using UDP P2P Reliable Distributed Broadcast scheme

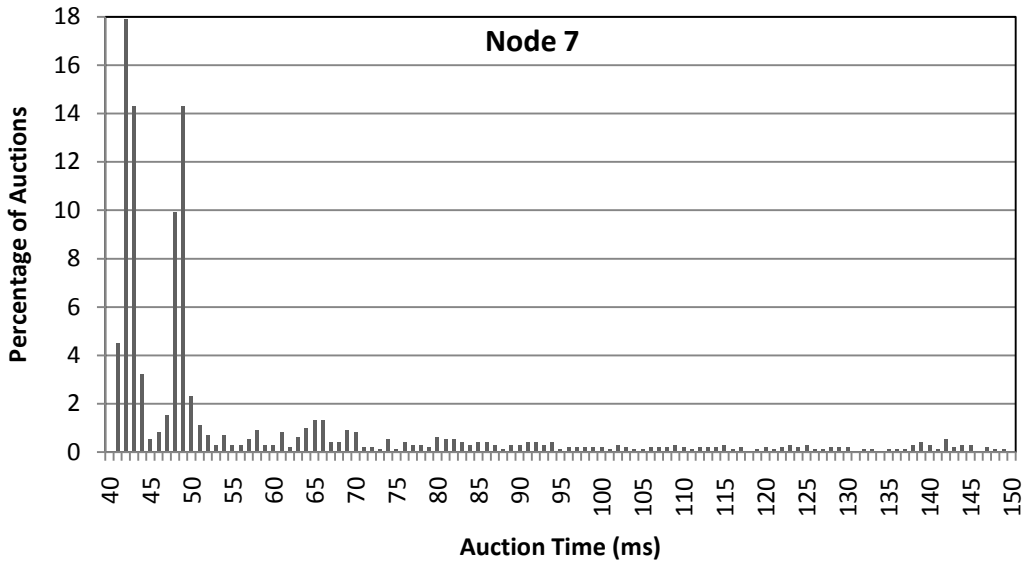

Figure 5.27: Distribution of Auction completion times of Node 7, when three nodes are mobile using UDP P2P Reliable Distributed Broadcast scheme

Figures Figure 5.25, Figure 5.26, andFigure 5.27 show the distribution of Auction completion times of nodes 2, 3, and 7 respectively i.e., these are the three nodes that are mobile. Euclidean distance is used to measure the similarity between the histograms. Tables (Table 5.6 &Table 5.7) below show the Average Auction Time and Standard Deviation of each node i.e., nodes 2, 3, and 7, and Euclidean distance between nodes (2, 3), nodes (3, 7), and nodes (2, 7). While comparing Average Auction Time and Standard Deviation of each node and Euclidean distance between nodes (2, 3), nodes (3, 7), and nodes (2, 7) we can say that histograms 3 and 2 have similar distance to that of 7.

Node	Average Auction Time (ms)	Standard Deviation
Node 2	57	24.6
Node 3	57.4	25
Node 7	56.1	24

Table 5.6: Average Auction Time and Standard Deviation of nodes 2, 3 and 7

d1(2&3)	d2(2&7)	d3(3&7)
51.7	81	85

Table 5.7: Euclidean distance between nodes (2, 3), nodes (3, 7), and nodes (2, 7)

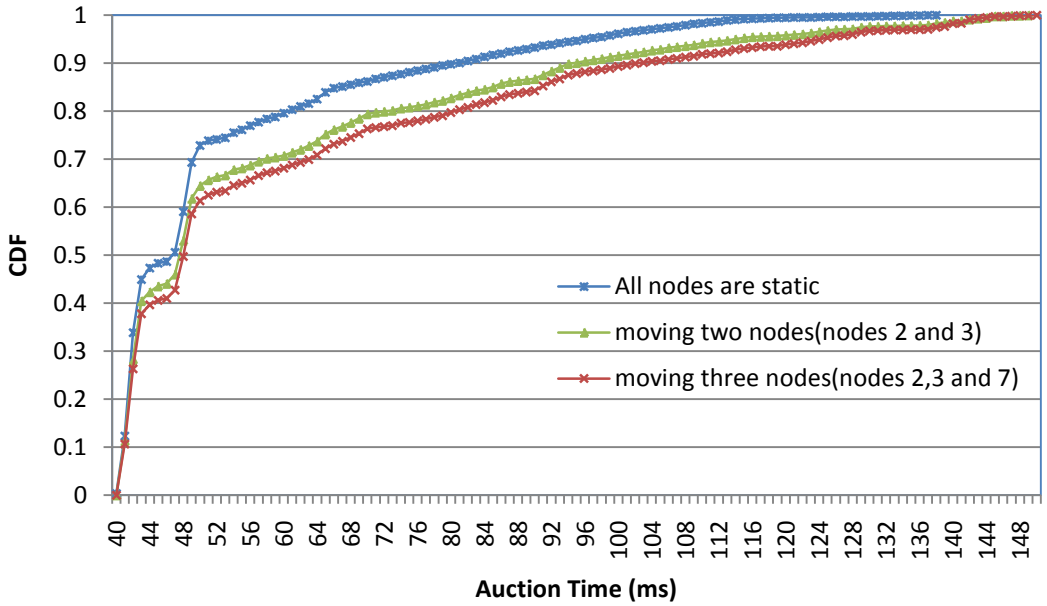

Figure 5.28: CDF's of Auction times of three different scenarios using UDP P2P Reliable Distributed Broadcast scheme

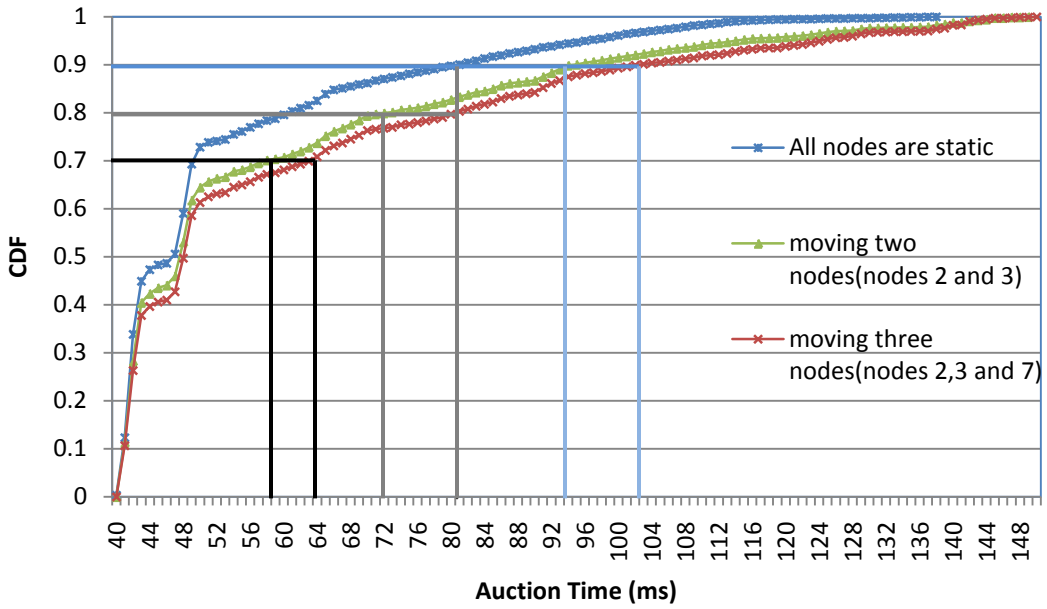

Figure 5.29: CDF's of Auction times of three different scenarios using UDP P2P Reliable Distributed Broadcast scheme

The CDF of Auction times of three different scenarios i.e., CDF of Auction times when all the nodes are static, CDF of Auction times when two nodes are mobile, and CDF of Auction times when three nodes are mobile using UDP P2P Reliable Distributed Broadcast scheme is shown in the above Figure 5.28. When moving the nodes we notice an increase in disparity, as the number of nodes moving increases, the disparity is also increasing. It is clear that, 70% of the Auctions are completed in less than 49ms when all the nodes are static, whereas it is slightly increases to 57 ms, when two nodes are mobile. When three nodes are mobile, the 70% of auctions are completed within 40-63ms. Moreover, 80% of the Auctions are completed in less than 61ms when all the nodes are static, whereas it slightly increases to 71 ms, when two nodes are mobile. When three nodes are mobile, 80% of Auctions are within range 40-80ms. 10% of the Auctions are within the range 82-149ms when all the nodes are static, whereas it slightly decreases to

93-149ms when two nodes are mobile. When three nodes are mobile, 10% of Auctions are within the range 103-149ms.

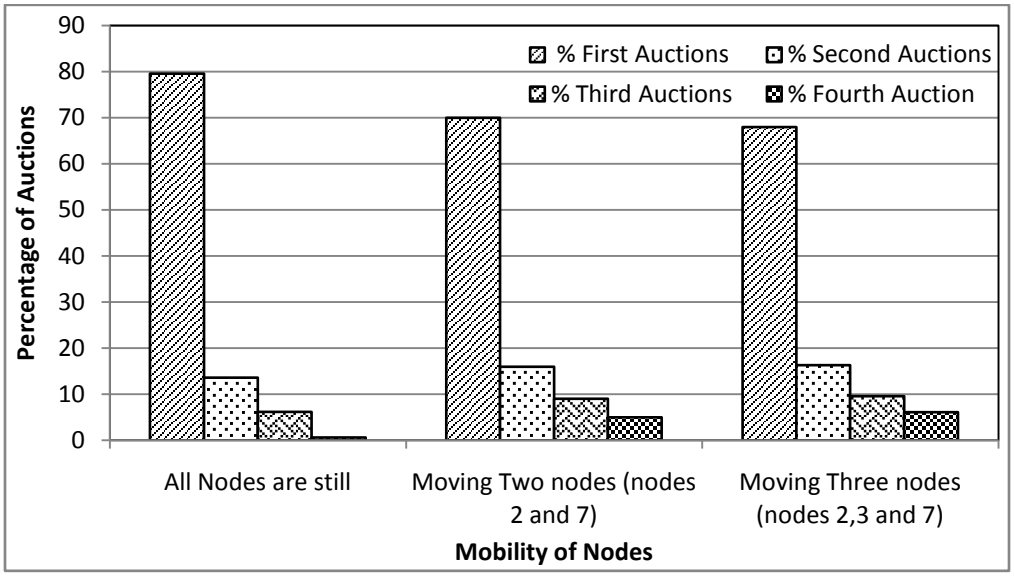

Figure 5.30: Reliability of UDP P2P Reliable Distributed Broadcast scheme with three different scenarios

Scenario	Average Auction Time (ms)	Standard Deviation (S.D)
All Nodes are static	53	17.7
Two Nodes are mobile	58.3	24.4
Three Nodes are mobile	60.7	26.6

Table 5.8: Summary of Average Auction Time (ms) and Standard Deviation of each scenario using UDP P2P Reliable Distributed Broadcast Scheme

The UDP P2P Reliable Distributed Broadcast scheme appears to be reliable, as all the experienced auctions among seven Stargate nodes are completed using only four auctioning steps. Figure 5.30 shows the Percentage of Auctions by all the nodes in three different scenarios, i.e. Percentage of Auctions completed in the First attempt, the Second

attempt, the Third attempt, or the Fourth attempt. One needs four auctions to make sure that all the nodes of our Ad-hoc network have successfully replied back, although, when 30% of the nodes are mobile (two nodes are moving), the Average auction time is increased by only 58.3ms with S.D = 24.4, and when 40% of the nodes are mobile (three nodes are moving), the Average auction time is increased by only 60.7ms with S.D = 26.6.

5.5 Comparison of Proposed & Previous Schemes

In this section, we present the comparison of Auction time (Response time) of each scheme i.e., the proposed schemes as well as previous schemes found in literature.

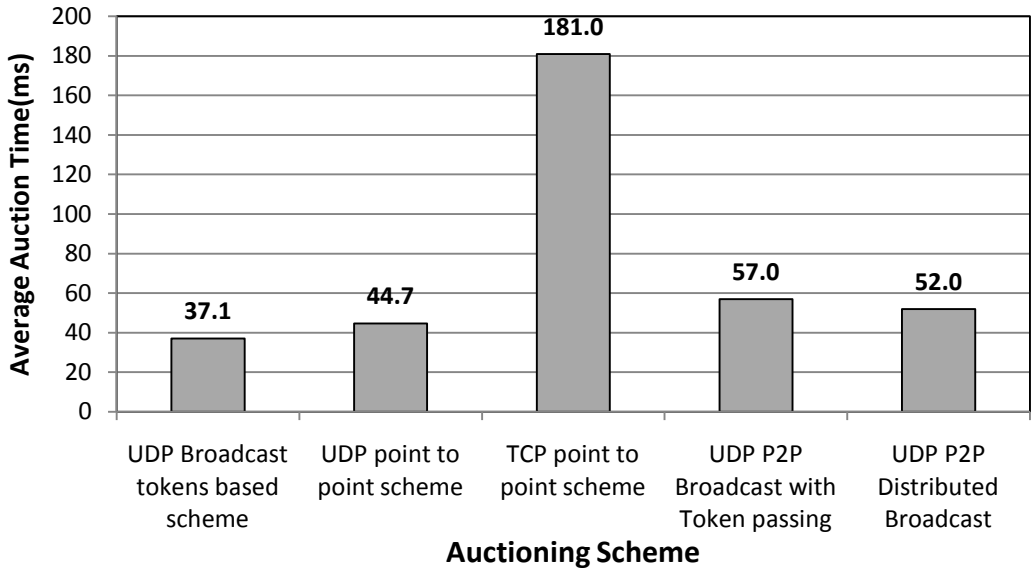

Figure 5.31: Comparison of Average Response Time per Auction by Each Scheme

Figure 5.31 shows the comparison of Average Response Times of each scheme. The average response time is calculated by taking into account 1000 auctions by each scheme. The first two schemes are UDP based and these schemes are not peer to peer schemes,

i.e. in these schemes there is a head node which is responsible for generating auctions and measuring the response time of each auction [3]. The Average response times for UDP Broadcast with token passing scheme, and for UDP point to point schemes are 37ms and 45ms respectively. The third scheme shown is also found in the literature, and is totally TCP based. In this technique, the head node is responsible for generating auctions and measuring the response time of each auction. The Average response time of the TCP point to point scheme is 181ms. The last two schemes are our proposed schemes; these schemes are totally peer to peer based, unlike the other schemes which are not peer to peer schemes. In these schemes there is no head node which is responsible for generating the auctions. Each node acts like a peer, i.e. any node can generate an auction and measure the Auction time. The Average response times of UDP P2P Reliable Broadcast with Token Passing scheme and UDP P2P Reliable Distributed Broadcast scheme are found to be 57ms and 52ms, respectively. The Average Response time of our proposed schemes is slightly greater than the previously proposed UDP schemes. This is because in the previous schemes there is only one node (head node) which generates the auctions and other nodes know to whom they should reply back with an ACK, whereas in our proposed schemes any node can generate the auction, i.e. the same code is run on all the peers, and each peer has to know to whom they have to reply back with an ACK. This requires extracting the source IP address and data from the received packet. As a result of which we are having the Average response time to be slightly greater than the other UDP schemes.

5.6 Comparison of Power Consumption of Proposed & Previous Schemes

In this section we present the comparison of average power consumption for an auction by each communication scheme i.e., the proposed schemes as well as previous schemes found in the literature.

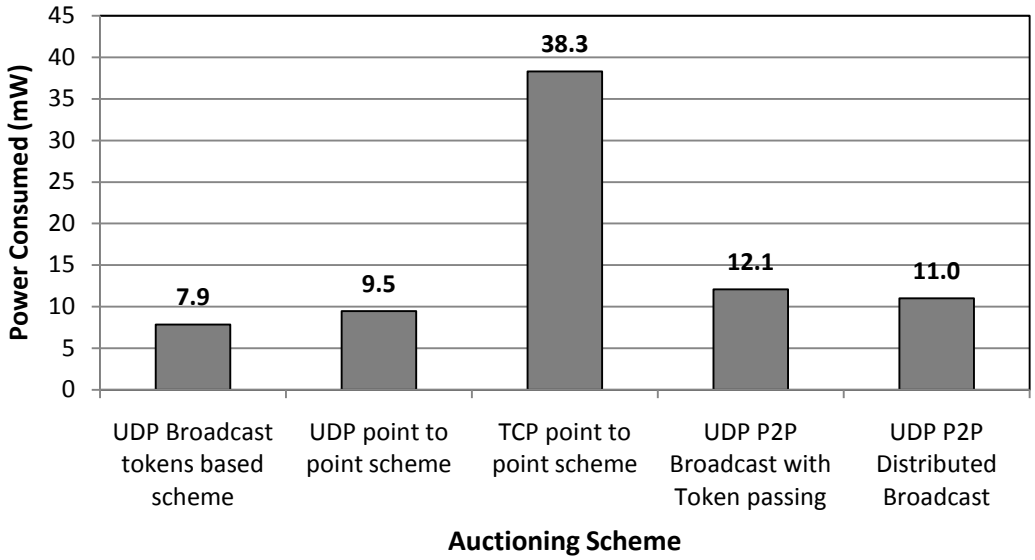

Figure 5.32: Comparison of Average Power consumption per Auction by each scheme

Measurements of energy consumption in a visual sensor network are reported in [28] for CPU processing, flash memory access, image acquisition, and communication, characterized for different hardware states like sleep, idle, transmitting, and receiving, and webcam on/off. Figure 5.32 shows the comparison of power consumption by each scheme. It reveals that wireless communication consumes a lot of energy. Direct measurements of the current during the idle phase, communication phase, and computation phase of the Stargate module gives 0.37A, 0.46A, and 0.52A, respectively. We use current = 0.46A to measure power consumption of each scheme, because we

measured the time taken to complete an auction in the communication phase of the auction. Measurement of power consumption using the UDP Broadcast with token passing scheme, UDP Point to Point scheme, TCP point to point scheme, UDP P2P Reliable Broadcast with Token Passing scheme, and UDP P2P Reliable Distributed Broadcast scheme reveals that the average power consumptions per auction are 38.3 mW, 9.4 mW, 7.9 mW, 12.069mW, and 10.99mW respectively. We used the following formula to calculate the power consumption by each scheme:

$$\text{Power} = \text{Current} \times \text{Current} \times \text{Time (Watt)} \quad (5.2)$$

Table 5.9 summarizes the average completion times (Average Auction time taken by each scheme) and the power consumption by each communication scheme.

Scheme	**Average time taken by one auction in milliseconds**	**Power Consumed (using current x current x time) mW**
UDP Broadcast tokens based scheme	37	7.8
UDP point to point scheme	44.6	9.4
TCP point to point scheme	180.9	38.2
UDP P2P Reliable Broadcast with Token passing scheme	57	12
UDP P2P Reliable Distributed Broadcast scheme	51.9	10.9

Table 5.9: Summary of Average Auction completion times and their corresponding power consumption

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this chapter, we summarize the thesis work and its contributions to the reliability of the broadcast in IEEE 802.11 networks. Section 6.1 provides the conclusion of the thesis. Section 6.2 highlights the contributions of the research in the area of Reliable Broadcast in Ad-hoc WLAN using IEEE 802.11b. Future research based on this thesis and general directions in related areas are described in section 6.3.

6.1 Conclusion

In this thesis, we presented the design and implementation of UDP P2P Reliable Broadcast with Token Passing protocol and UDP P2P Reliable Distributed Broadcast protocol for Ad-hoc WLANs. These schemes use an imperative Poll-based communication like in SNMP. Each peer node runs two threads: (1) a communication thread (TC), and (2) a processing thread (TP). We described how these protocols operate within each thread. We presented experimental data for assessing the degree of reliability of the proposed protocols altogether with overall auctioning times. UDP P2P Reliable Broadcast with Token Passing protocol appears to be reliable as all experienced auctions among 7 stargate nodes are completed using only three auctioning steps. The first

auctioning step was successfully covered by all the nodes in 80% of the test cases, 12% of the test cases required a second auctioning step to cover all the nodes, and remaining 8% of the test cases required a third auctioning step to cover all the 7 stargate nodes. UDP P2P Reliable Distributed Broadcast protocol for WLAN are appears to be reliable, as all experienced auctions among 7 stargate nodes are completed using only four auctioning steps. All nodes in 82% of the test cases successfully covered the first auctioning step. Moreover, 12% of the test cases required a second auctioning step to cover all the nodes, 5.5% of the test cases required a third auctioning step to cover all the nodes, and 0.5% of the test cases required a fourth auctioning step to cover all the nodes. Slight variations in the average auction time when devices are mobile. The general observations from the evaluation of both the protocols are:

1) Symmetric code in all the nodes (peer to peer communication).
2) Response times are comparable to the UBTP auction scheme operated at head node.
3) Improved degree of reliability; at most 3 steps for seven nodes for UDP P2P Reliable Broadcast with Token Passing scheme, and at most 4 steps for seven nodes for UDP P2P Reliable Distributed Broadcast scheme.
4) Comparable power consumption to simple UBTP auction scheme.
5) Slight variations in the average auction times when devices are mobile.
6) Symmetric performance pattern of all the nodes in both the proposed protocols.

6.2 Contributions

The contributions of this thesis are:

- The design and implementation of two P2P Communication Protocols for IEEE 802.11b WLAN consisting of 7 Stargate embedded systems (7 nodes). The algorithms are deadlock-free i.e., while communicating with the other peer nodes the system should not get stuck.
- Evaluation of their performance such as Time taken to complete one Auction, measured reliability and power consumption by each Broadcasting scheme.
- Comparison of the proposed schemes with the other available schemes in literature in terms of auction times and power consumption by each scheme.

6.3 Future work

The new algorithms that we introduced in this research work showed improved performance over the broadcast mechanism of the IEEE 802.11 standard. In this section, we are going to show some of the improvements that could be done to these algorithms. Leaving these improvements as a future work, we believe we can get greater benefit from the proposed algorithms.

We have conducted the experiments on 7 stargate devices in an indoor environment. Experiments can be conducted outdoors, and Average response times (Average auction time), reliability and impact of mobility on each proposed scheme can be measured, and the effect of indoor to outdoor can be seen.

We have designed and implemented the proposed algorithms for single collision domain i.e. for single hop WLANs; these algorithms can be extended to multiple collision domains or for multiple hop WLANs.

As we have used static timing in each proposed algorithms, this timing can be made adaptive. The future work can be to study an adaptive version of the proposed protocols, which uses the knowledge of the timing from past history for iteratively converging to more adapted timing. This work is just the tip of an ice berg, a lot needs to be explored, and this thesis work will be a perfect starting point for any future research in this area.

APPENDIX

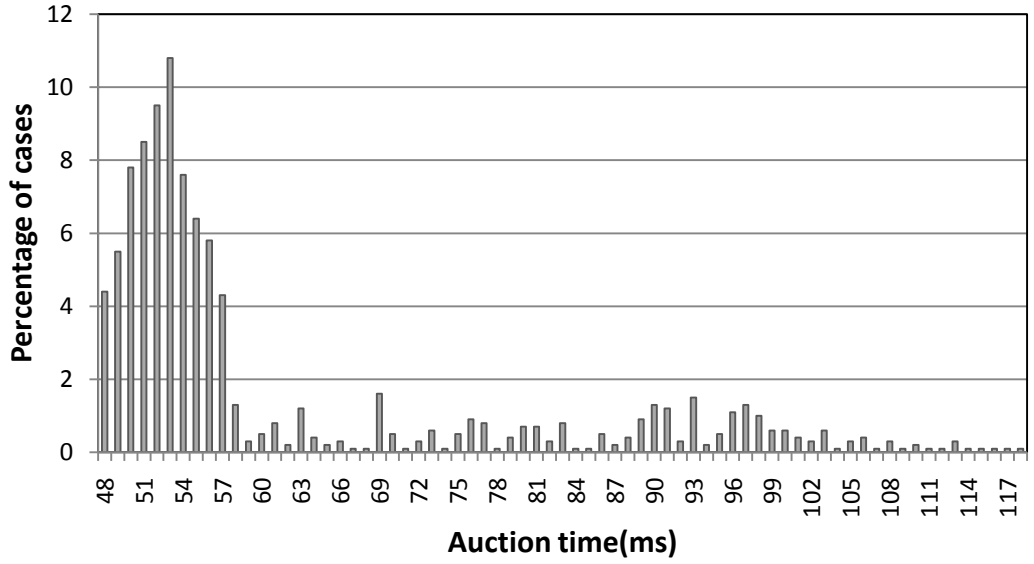

Figure A.1: Distribution of Completion times of Node1 (UDP P2P Reliable Broadcast with Token Passing scheme)

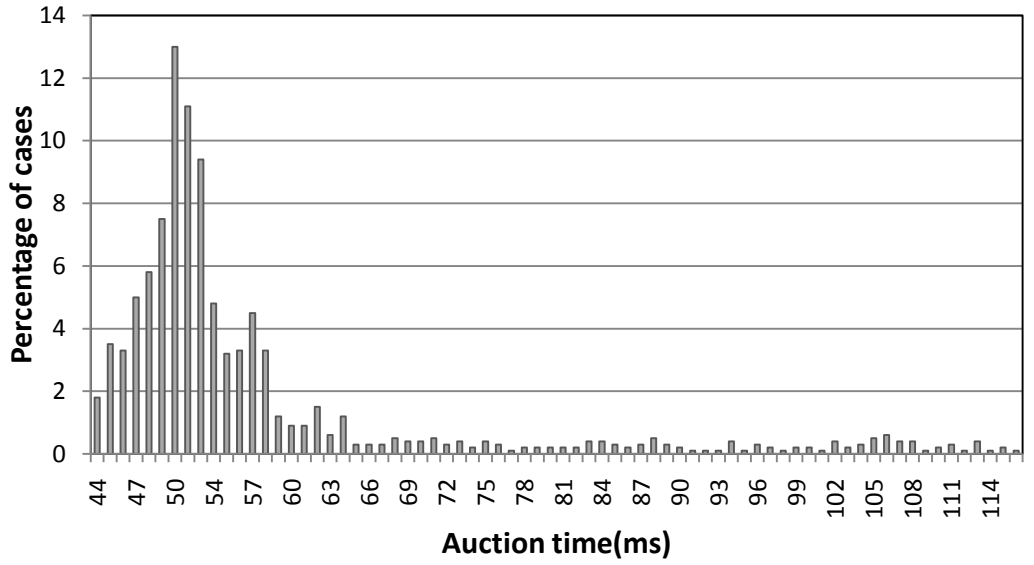

Figure A.2: Distribution of Completion times of Node2 (UDP P2P Reliable Broadcast with Token Passing scheme)

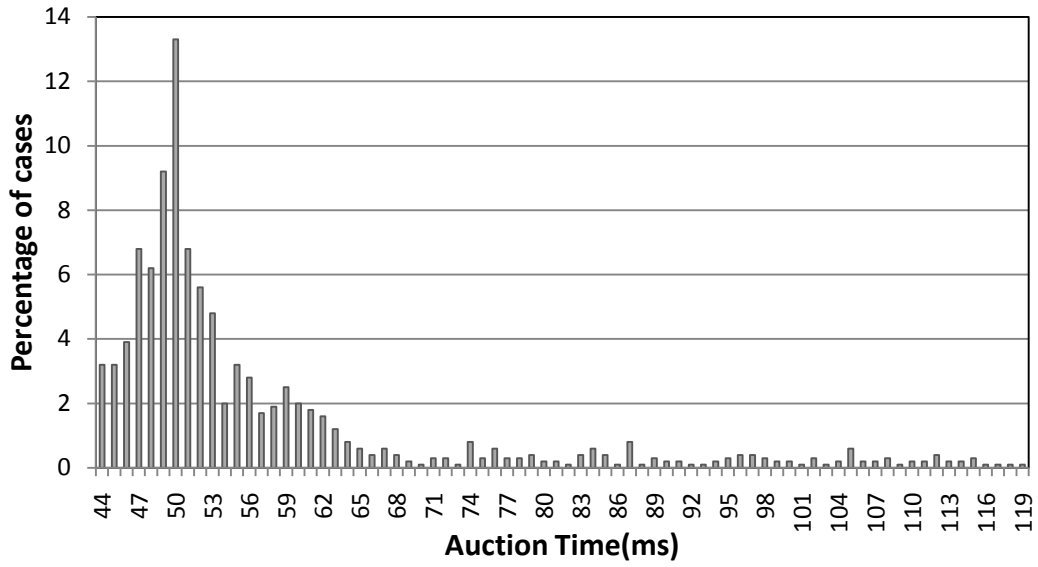

Figure A.3: Distribution of Completion times of Node3 (UDP P2P Reliable Broadcast with Token Passing scheme)

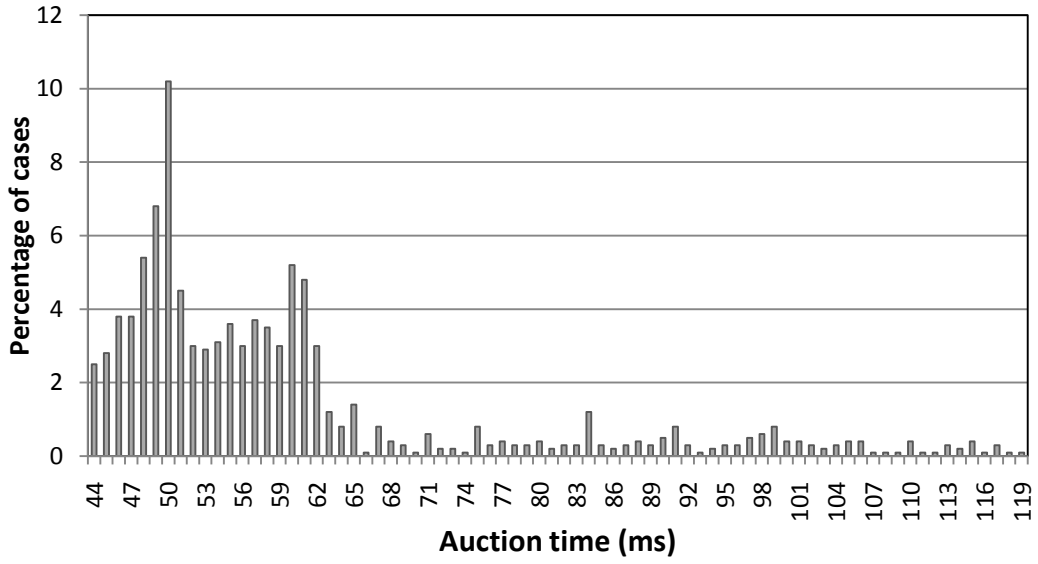

Figure A.4: Distribution of Completion times of Node4 (UDP P2P Reliable Broadcast with token passing scheme)

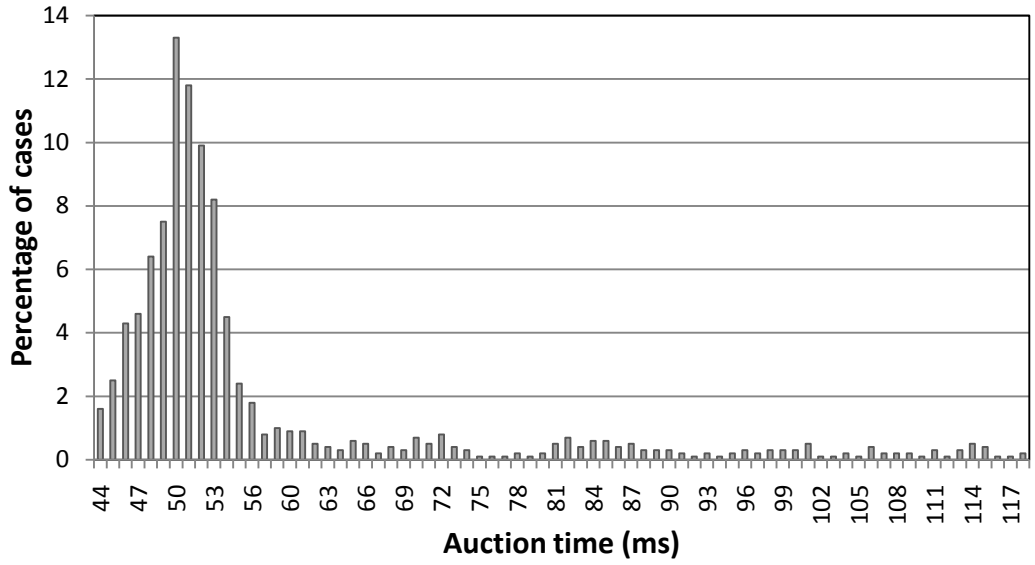

Figure A.5: Distribution of Completion times of Node5 (UDP P2P Reliable Broadcast with Token Passing scheme)

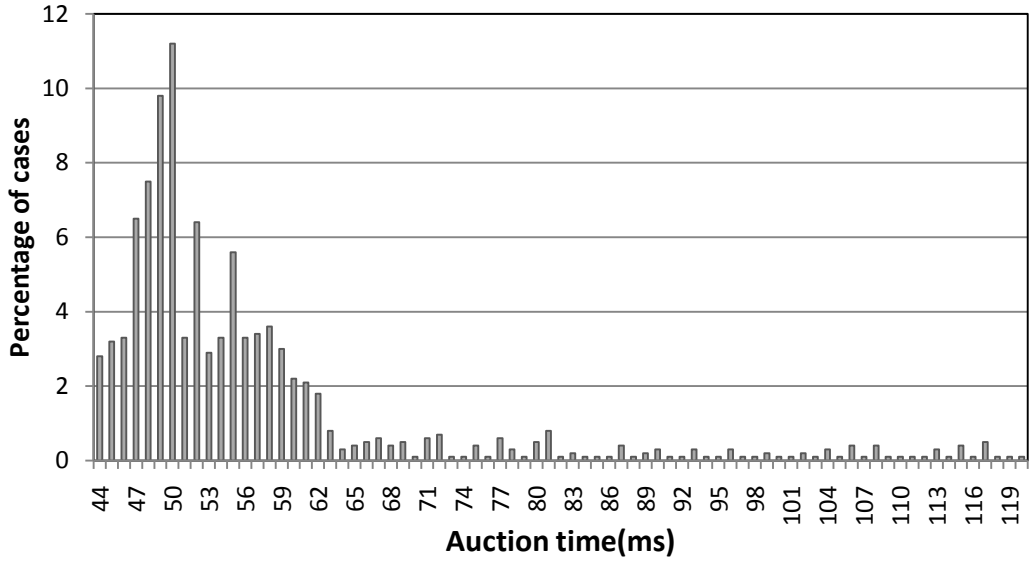

Figure A.6: Distribution of Completion times of Node6 (UDP P2P Reliable Broadcast with Token Passing scheme)

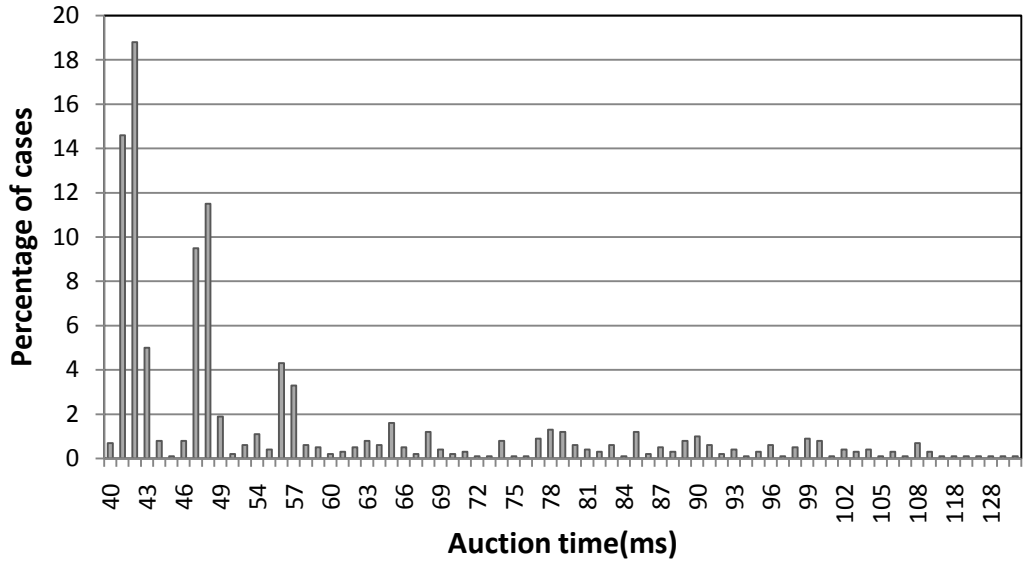

Figure A.7: Distribution of Completion times of Node1 (UDP P2P Reliable Distributed Broadcast scheme)

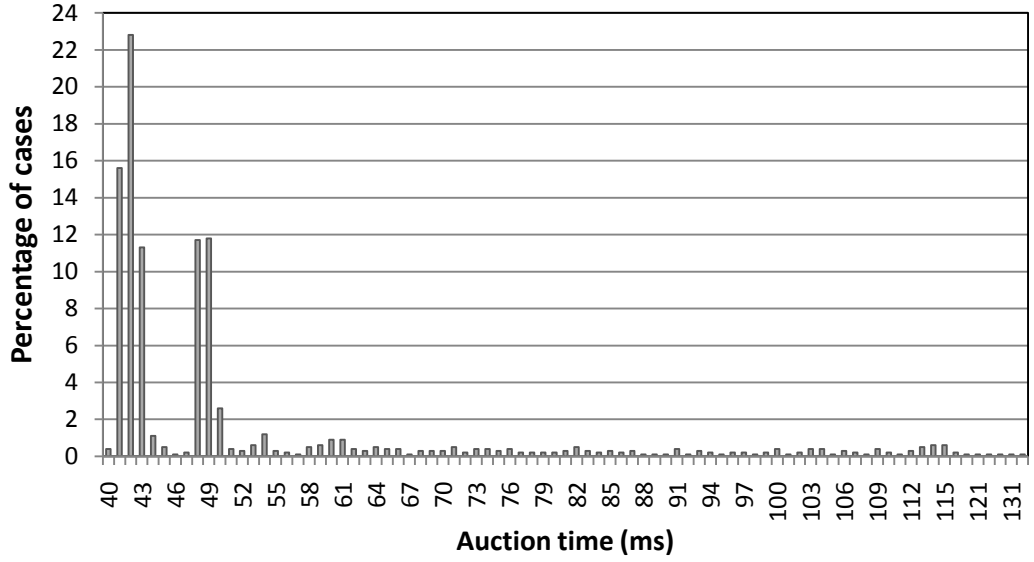

Figure A.8: Distribution of Completion times of Node2 (UDP P2P Reliable Distributed Broadcast scheme)

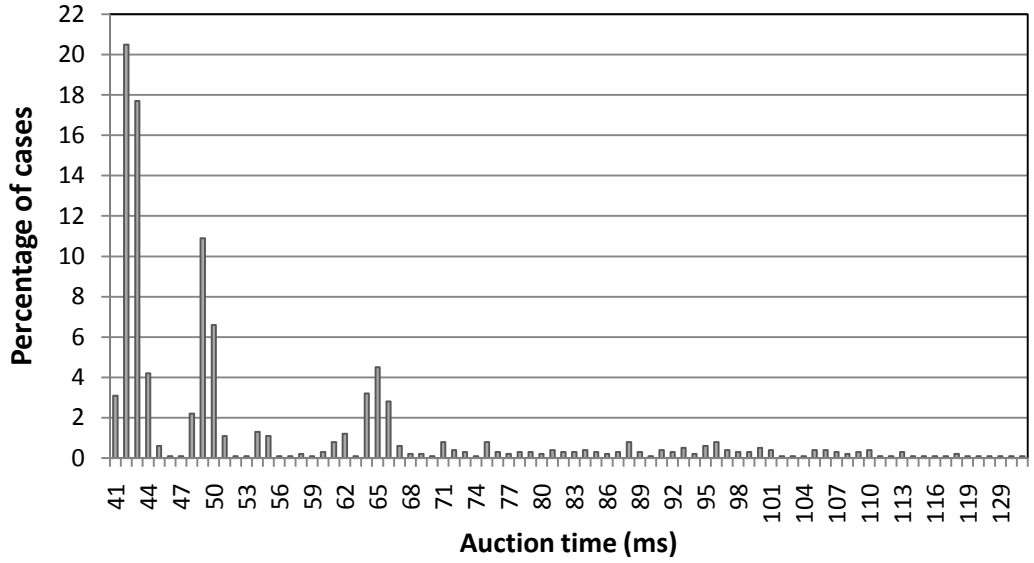

Figure A.9: Distribution of Completion times of Node4 (UDP P2P Reliable Distributed Broadcast scheme)

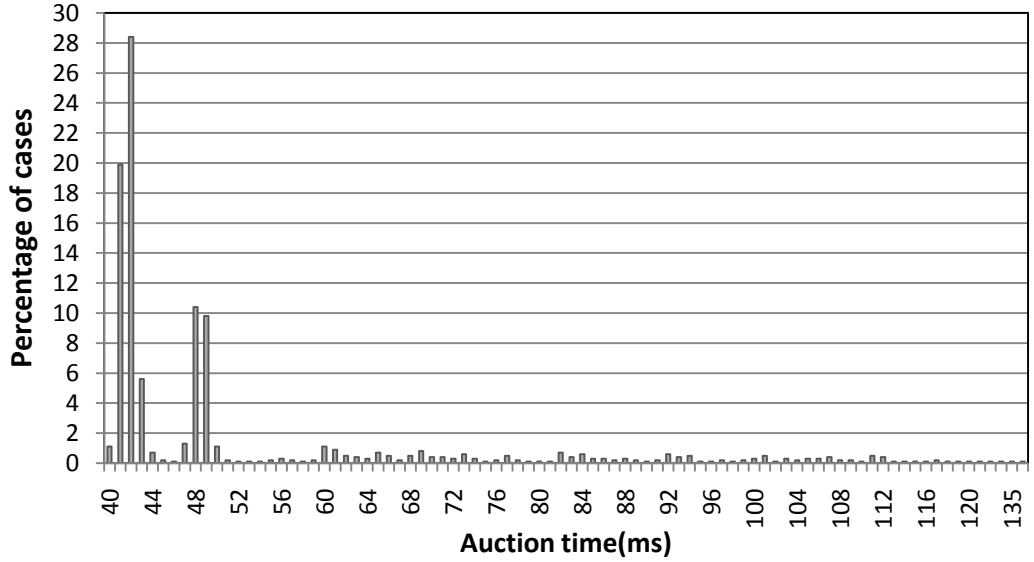

Figure A.10: Distribution of Completion times of Node5 (UDP P2P Reliable Distributed Broadcast scheme)

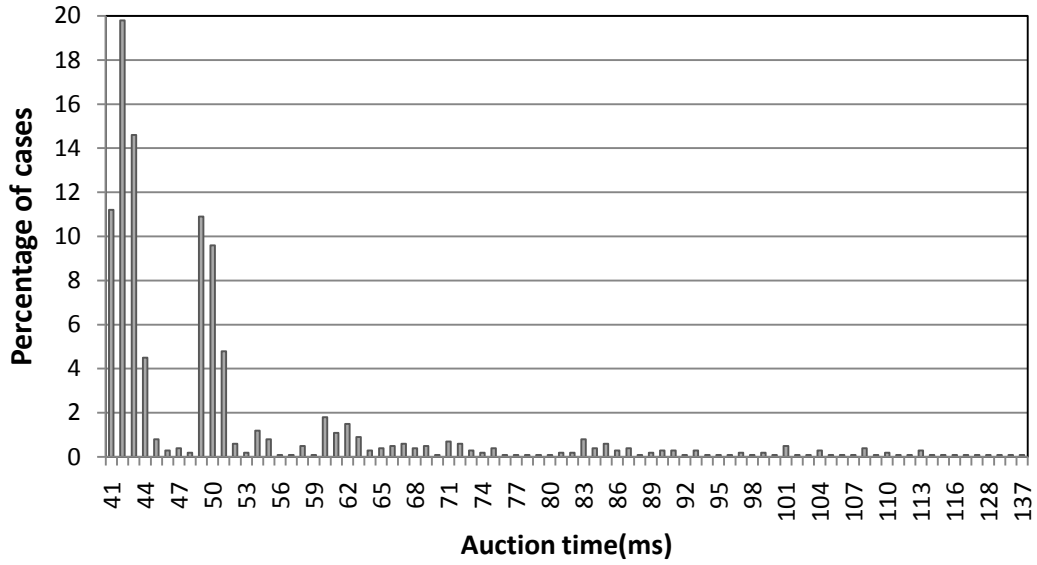

Figure A.11: Distribution of Completion times of Node6 (UDP P2P Reliable Distributed Broadcast scheme)

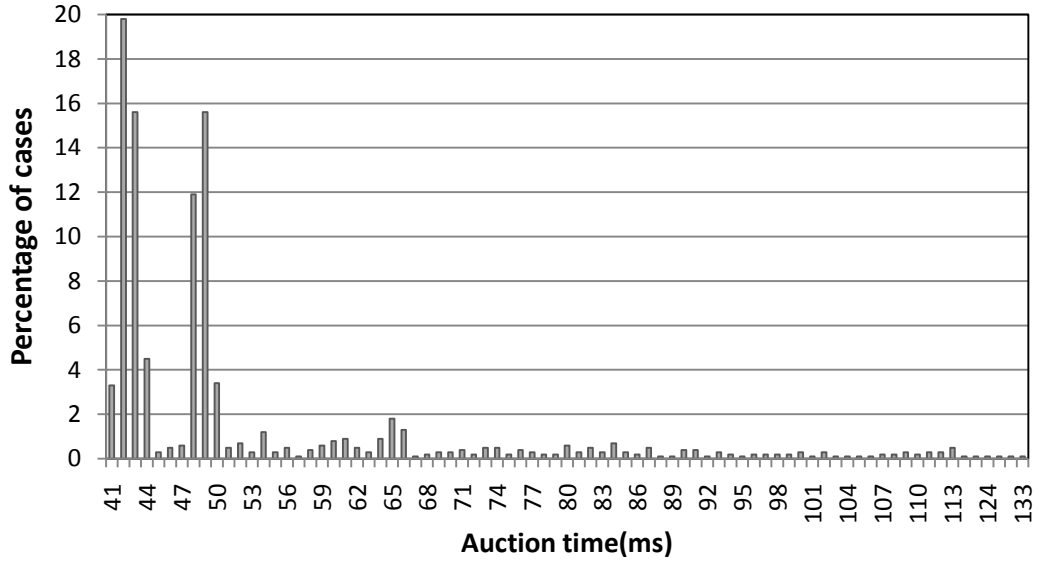

Figure A.12: Distribution of Completion times of Node7 (UDP P2P Reliable Distributed Broadcast scheme)

REFERENCES

[1] P. Lima, *et al.*, "A Functional Architecture for a Team of Fully Autonomous Cooperative Robots," in *RoboCup-99: Robot Soccer World Cup III, Lecture Notes in Computer Science*, ed: Springer Berlin / Heidelberg, pp. 37-98, 2000.

[2] B. van der Vecht and P. Lima, "Formulation and Implementation of Relational Behaviours for Multi-robot Cooperative Systems," in *RoboCup 2004: Robot Soccer World Cup VIII, Lecture Notes in Computer Science*, ed: Springer Berlin / Heidelberg, pp. 516-523, 2005.

[3] M. A. Al-Mouhamed and U. F. Siddiq, "Performance evaluation of auctions WLAN for RoboCup multi-robot cooperation," *IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*, pp. 610-615, 2009.

[4] Q. Ni, *et al.*, "A survey of QoS enhancements for IEEE 802.11 wireless LAN: Research Articles," *Wireless Communication & Mobile Computing*, vol. 4, pp. 547-566, 2004.

[5] "IEEE 802.11b Wireless LANs," www.3com.com/corpinfo/en_us/technology/tech_paper, 2001.

[6] F. Alizadeh-Shabdiz and S. Subramaniam, "Analytical models for single-hop and multi-hop ad hoc networks," *Mob. Netw. Appl.*, vol. 11, pp. 75-90, 2006.

[7] G. Anastasi, *et al.*, *IEEE 802.11 AD HOC Networks: Protocols, Performance, and Open Issues*: John Wiley & Sons, Inc., 2005.

[8] J. Villalón, *et al.*, *Efficient Joint Unicast/Multicast Transmission over IEEE 802.11e WLANs* vol. 284: Springer Boston, 2008.

[9] "IEEE 802.11 Wireless LAN Medium Access Control (MAC) and Physcal Layer (PHY) Specifications," *LAN MAN Standards Committee of the IEEE Computer Society,* June 1997.

[10] "IEEE 802.11 Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications," *International Standard for Information Technology*, 1999.

[11] W. Kiess and M. Mauve, "A survey on real-world implementations of mobile ad-hoc networks," *Ad Hoc Netw.*, vol. 5, pp. 324-339, 2007.

[12] L. Yang, *et al.*, "A Study Towards Reliability- and Delay-Critical Wireless Communication for RoboCup Robotic Soccer Application," *International Conference on Wireless Communications, Networking and Mobile Computing (WiCom)*, pp. 633-636, 2007.

[13] J. Kuri and S. K. Kasera, "Reliable Multicast in Multi-Access Wireless LANs," *Wireless Networks,* vol. 7, pp. 359-369, 2001.

[14] K. Tang and M. Gerla, "MAC reliable broadcast in ad hoc networks," *IEEE Military Communications Conference (MILCOM)*, pp. 1008-1013, 2001.

[15] S. Min-Te, *et al.*, "Reliable MAC layer multicast in IEEE 802.11 wireless networks," *International Conference on Parallel Processing*, pp. 527-536, 2002.

[16] S. Shiann-Tsong, *et al.*, "A highly reliable broadcast scheme for IEEE 802.11 multi-hop ad hoc networks," *IEEE International Conference on Communications (ICC)*, pp. 610-615, 2002.

[17] W. Xiaoli, *et al.*, "Reliable Multicast Mechanism in WLAN with Extended Implicit MAC Acknowledgment," *IEEE Vehicular Technology Conference (VTC)*, pp. 2695-2699, 2008.

[18] J. Tourrilhes, "Robust broadcast: improving the reliability of broadcast transmissions on CSMA/CA," *The Ninth IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1111-1115, 1998.

[19] X. Jiawei, *et al.*, "Improving the reliability of IEEE 802.11 broadcast scheme for multicasting in mobile ad hoc networks," *IEEE Wireless Communications and Networking Conference*, pp. 126-131, 2005.

[20] C. A. C. Parker and Z. Hong, "A Practical Implementation of Random Peer-to-Peer Communication for a Multiple-Robot System," *IEEE International Conference on Robotics and Automation*, pp. 3730-3735, 2007.

[21] R. O. LaMaire, *et al.*, "Analysis of a wireless MAC protocol with client-server traffic and capture," *IEEE Journal on Selected Areas in Communications*, vol. 12, pp. 1299-1313, 1994.

[22] I. R. Chen, *et al.*, "On failure recoverability of client-server applications in mobile wireless environments," *IEEE Transactions on Reliability*, vol. 54, pp. 115-122, 2005.

[23] B. Jeppe, *et al.*, "A Uniform Publish-Subscribe Infrastructure for Communication in Wireless Mobile Environments," *6th International Conference on ITS Telecommunications Proceedings* pp. 1145-1150, 2006.

[24] Y. Wen-Jun, *et al.*, "Decomposition of Subscription Based Publish-Subscribe Middleware for Wireless Sensor Networks," *International Workshop on Intelligent Systems and Applications (ISA)*, pp. 1-4, 2009.

[25] R. Amin, *et al.*, "Analyzing performance of ad hoc network mobility models in a peer-to-peer network application over mobile ad hoc network," *International Conference On Electronics and Information Engineering (ICEIE)*, pp. 344-348, 2010.

[26] L. Eng Keong, *et al.*, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys & Tutorials*, vol. 7, pp. 72-93, 2005.

[27] "Stargate users manual - Stargate Developers Guide," www.xbow.com/Support/Support_pdf_files/Stargate_Manual.pdf, January 2006.

[28] C. B. Margi, *et al.*, "Characterizing energy consumption in a visual sensor network testbed," *2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, pp. 334-339, 2006.

VITA

- Name: Irfan Ali Khan
- Date of Birth: 25th August 1984
- Nationality: Indian
- Received Bachelor of Technology (B.Tech) with honors in Electronics and Communications Engineering from Jawaharlal Nehru Technological University, Hyderabad, India, 2007.
- Joined Computer Engineering Department at King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia, Research Assistant in October 2008.
- Received Master of Science (M.S) in Computer Engineering from KFUPM in January 2011.
- Present Address: Bldg.802, Room 303, KFUPM No. 8586, Dhahran 31261, Saudi Arabia.
- Permanent Address: H.No: 8-4-549/5/8, Sultan Nagar, Erragadda, Hyderabad, India, Pin Code: 500018.
- E-mail: irfankhanin2005@gmail.com
- Mobile Number: 00966530058907